Universidad de Alcalá
Escuela Politécnica Superior

Máster Universitario en Ingeniería Industrial

Trabajo Fin de Máster

Smart HMI for an autonomous vehicle

Autor: Javier Araluce Ruiz

Tutor: D. Luis Miguel Bergasa Pascual

2020

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Máster Universitario en Ingeniería Industrial

Trabajo Fin de Máster

Smart HMI for an autonomous vehicle

Autor: Javier Araluce Ruiz

Director: D. Luis Miguel Bergasa Pascual

Tribunal:

Presidente: Dña. Elena López Guillén

Vocal 1º: D. Ignacio Parra Alonso

Vocal 2º: D. Luis Miguel Bergasa Pascual

Calificación: .....

Fecha: .....

Acknowledgements

Este trabajo final de Máster pone fin a dos años en los que se podría decir que he encontrado el camino a seguir laboralmente a base de dedicar largas jornadas en el interior de ese edificio compaginando en la medida de lo posible mi actividad investigadora con el máster. Pero es que, si quieres algo, algo te cuesta.

En primer lugar, quería agradecer este trabajo a mi tutor y supervisor D. Luis Miguel Bergasa Pascual por guiarme a la hora de realizar este trabajo y en especial por proponerme retos continuamente los cuales yo no me había llegado a plantear, los cuales han formado parte de este trabajo y confío que sigan siendo en los trabajos futuros.

A mis compañeros de laboratorio: Carlos, Javi, Óscar, Alex, Rodri y Felipe que, aunque este año no nos hemos visto mucho, hemos tenido contacto diario para que cada uno pueda llegar más lejos de lo que inicialmente se había propuesto en su investigación gracias a una ayuda constante, consiguiendo no solo crecer profesionalmente, sino también personalmente.

A mis compañeros de Máster, que han permitido que pueda realizar mi actividad investigadora además de sacarme el Máster, realizando trabajos en horarios que quizás no eran los más ideales para ellos y siguiendo una organización algo estricta.

Y por último a mi familia que no solo ha tenido que aguantarme estos meses más tiempo en casa, sino que se han visto forzados a tener que realizar diferentes pruebas para la realización de este trabajo, las cuales han realizado pese a mis criticas a veces no tan constructivas en la forma de desarrollar este, jejeje.

Y no me podía olvidar de todos los anónimos que me han ayudado en la realización de este trabajo gracias a sus aportes a los que he podido acceder gracias a Google, Github, Stackoverflow ...

Resumen

El presente trabajo expone la arquitectura diseñada para la implementación de un HMI (Human Machine Interface) en un vehículo autónomo desarrollado en la Universidad de Alcalá. Este sistema hace uso del ecosistema ROS (Robot Operating System) para la comunicación entre los diferentes modulos desarrollados en el vehículo.

Además se expone la creación de una herramienta de captación de datos de conductores haciendo uso de la mirada de este, basada en OpenFace [1], una herramienta de código libre para análisis de caras. Para ello se han desarrollado dos métodos, uno basado en un método lineal y otro usando técnicas del algoritmo NARMAX. Se han desarrollado diferentes test para demostrar la precisión de ambos métodos y han sido evaluados en el dataset de accidentes DADA2000 [2].

Palabras clave: HMI, atención del conductor, escenarios con accidentes, estimación de la mirada,mapa de calor, visión artificial.

Abstract

This works presents the framework that composed the HMI (Human Machine Interface) built in an autonomous vehicle from University of Alcalá. This system has been developed using the framework ROS (Robot Operating System) for the communication between the different sub-modules developed on the vehicle.

Also, a system to obtain gaze focalization data from drivers using a camera is presented, based on OpenFace [1], which is an open source tool for face analysis. Two different methods are proposed, one linear and other based on NARMAX algorithm. Different test has been done in order to prove their accuracy and they have been evaluated on the challenging dataset DADA2000 [2], which is composed by traffic accidents.

Keywords: HMI, driver attention, accidental scenarios, gaze estimation, heat map, computer vision.

Extended Abstract

Autonomous vehicles represents the near future of the automobile industry and this improvement of the vehicles is going to be lead by Deep Learning. Deep learning needs a great amount of data to work and this data has to be validated by a human. Also autonomous vehicles will not appear all of a sudden, there will be a progressive change.

During this change autonomous vehicles will share the roads with human drivers and also some autonomous vehicles will be driven by humans which will need information about the decisions that the car is making.

This is the place where this Master thesis takes part. This works presents the framework that composed the HMI (Human Machine Interface) built in an autonomous vehicle from University of Alcalá. This system has been developed using the framework ROS (Robot Operating System) for the communication between the different sub-modules developed on the vehicle.

Two different HMIs have been developed, one for displaying information to the driver as can be velocity, battery level, D-GPS information, car state and the environment perceived by the vehicle.

Also, other screen, which can be controlled with the hand has been built on the car in order to change the navigation of the autonomous vehicle, selecting the paths to follow.

Understanding how an autonomous vehicle takes decisions while it is driving can be the key to achieve the total autonomy. So in order to get this, this works presents the development of a system to obtain gaze focalization data from drivers using a camera, based on OpenFace [1], which is an open source tool for face analysis.

Obtaining gaze focalization is an expensive and intrusive task for drivers because actual gaze-trackers are based on intrusive and expensive systems, which differs from what the authors consider a naturalistic driving.

This works presents two different methods, one linear and other based on NARMAX algorithm. Different test has been done in order to prove their accuracy and they have been evaluated on the challenging dataset DADA2000 [2] which is composed by traffic accidents.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Autonomous Vehicles (AVs) are transport systems capable of sensing their environment and operating without human involvement. A human passenger is not required to take control of the vehicle at any time, nor is a human passenger required to be present in the vehicle at all. An autonomous car can go anywhere a traditional car goes and do everything that an experienced human driver does.

AVs combine a variety of sensors to perceive their surroundings, such as radar, LIDAR, sonar, GPS, odometry and inertial measurement units. Advanced control systems interpret sensory information to identify appropriate navigation paths, as well as obstacles and relevant traffic sings.

Why Autonomous cars are the future of mobility? The scenarios for convenience and quality-of-life improvements are limitless. The elderly and the physically disabled would have independence. If your kids were at summer camp and forgot their bathing suits and toothbrushes, the car could bring them the missing items. You could even send your dog to a veterinary appointment. The project on which this work has been done is focused on an autonomous car to elderly as will be explained in next sections.

The Society of Automotive Engineers (SAE) currently defines 6 levels of driving automation ranging from Level 0 (fully manual) to Level 5 (fully autonomous) as can be seen on Figure 1.1.1. These levels have been adopted by the U.S. Department of Transportation.

This works describes the process to build an HMI (Human Machine Interface) for a level 3 autonomous vehicle and the development of a focalization tool in order to achieve the attention model generated by the driver while he is driving.

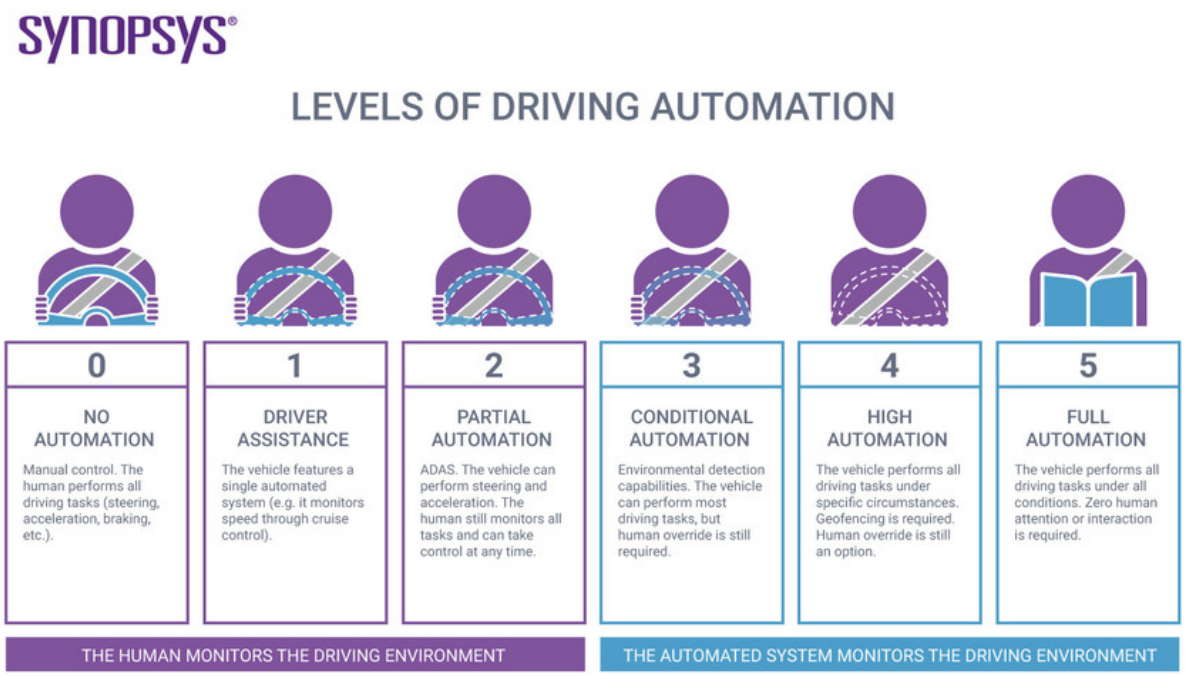


Figure 1.1.1: Levels of driving automation.

1.1.1 Levels of driving automation.

1.1.1.1 Level 0 (No Driving Automation)

Most vehicles that currently are on the road are Level 0: manually controlled. The human makes all the task to drive the car, although there may be systems on board to help the driver, as can be the emergency braking system (ABS).

1.1.1.2 Level 1 (Driver Assistance)

Lowest level of automation. The vehicle features a single automated system for driver assistance, such as steering or accelerating (cruise control). Adaptive Cruise Control (ACC), where the vehicle can be kept at a safe distance behind the next car, qualifies as Level 1 because the human driver monitors the other aspects of driving such as steering and braking the car only keeps the velocity to be at a safety distance from the ahead vehicle.

1.1.1.3 Level 2 (Partial Driving Automation)

This means Advanced Driver Assistance Systems (ADAS). The vehicle can control both steering and accelerating/braking. Here the automation falls short of self-driving because

a human sits in the driver's seat and can take control of the car at any time. Tesla Autopilot and Cadillac (General Motors) Super Cruise systems both qualify as Level 2.

1.1.1.4 Level 3 (Conditional Driving Automation)

The jump from Level 2 to Level 3 is substantial from a technological perspective, but subtle if not negligible from a human perspective.

Level 3 vehicles have "environmental detection" capabilities and can make informed decisions for themselves, such as an overtaking maneuver. The main difference is a decision-making layer which is in charge of the vehicle instead of the human. But they still require human override. The driver must remain alert and ready to take control if the system is unable to execute the task. So, the car can overtake other cars and take decisions of the driving procedure without being approved by the driver, who has to be alert because the car can pass him the control in every moment.

Almost two years ago, Audi [1] (Volkswagen [2]) announced that the next generation of the A8, their flagship sedan, would be the world's first production Level 3 vehicle. It launched the Traffic Jam Pilot, which combines a LIDAR scanner with advanced sensor fusion and redundant processing power to work even a component fails.

However, while Audi was developing their marvel of engineering, the regulatory process in the U.S. shifted from federal guidance to state-by-state mandates for autonomous vehicles. So for the time being, the A8L is still classified as a Level 2 vehicle in the United States and will ship without key hardware and software required to achieve Level 3 functionality. In Europe, however, Audi will roll out the full Level 3 A8L with Traffic Jam Pilot (in Germany first).

1.1.1.5 Level 4 (High Driving Automation)

The main difference between Level 3 and Level 4 automation is that Level 4 vehicles can intervene if things go wrong or there is a system failure. In this sense, these cars do not require human interaction in most circumstances. However, a human still has the option to manually override. If the human is not able to operate the car, the car will stop when and where it can, to give control to the driver because it is not able to drive any more, at least in those circumstances, as can be weather conditions, road works, etc.

Level 4 vehicles can operate in self-driving mode. But until legislation and infrastructure evolves, they can only do so within a limited area (usually an urban environment where top speeds reach an average of 50 km/h). This is known as geofencing (virtual

[1] https://www.audi.com/
[2] https://www.volkswagen.com/

perimeter). As such, most current Level 4 vehicles are geared toward ridesharing. Some examples:

- NAVYA [3], a French company, is already building and selling Level 4 shuttles and cabs in the U.S. that run fully on electric power and can reach a top speed of 88 km/h.

Figure 1.1.2: NAVYA autonomous car.

- Alphabet's Waymo [4] recently unveiled a Level 4 self-driving taxi service in Arizona, where they had been testing driverless cars without a safety driver in the seat for more than a year and over 10 million miles.

- Canadian automotive supplier Magna [5] has developed technology (MAX4) to enable Level 4 capabilities in both urban and highway environments. They are working with Lyft [6] to supply high-tech kits that turn vehicles into self-driving cars.

- Volvo[7] and Baidu [8] announced a strategic partnership to jointly develop Level 4 electric vehicles that will serve the robotaxi market in China.

[3] https://navya.tech/fr/
[4] https://waymo.com/
[5] https://www.magna.com/home
[6] https://www.lyft.com/
[7] https://www.volvocars.com/es
[8] http://www.baidu.com/

Figure 1.1.3: Alphabet's Waymo autonomous car.

Figure 1.1.4: Magna autonomous car.

1.1.1.6 Level 5 (Full Driving Automation)

Level 5 vehicles do not require human attention "dynamic driving task" is eliminated. Level 5 cars won't even need steering wheels or acceleration and braking pedals because the vehicle will have all the systems required to drive in any conditions. They will be free from geofencing, able to go anywhere and do anything that an experienced human driver can do. Fully autonomous cars are undergoing testing in several pockets of the world, but none are yet available to the general public.

Figure 1.1.5: Volvo and Baidu autonomous car.

1.2 Historical context

The idea of autonomous vehicles [3] has captured human imagination since the 1930s and with the 2020 slated deadline to put autonomous vehicles cars on the streets of major cities in the US and other developed countries, stakes and motivation are definitely high.

Carnegie Mellon University was the first organization to present a truly autonomous car (circa 1980); however, DARPA's (Defense Advanced Research Projects Agency) self-driving cars competition of 2004, 2005 and 2007 was the beginning of the race to create the first road autonomous vehicle. DARPA's challenge to university students and private contractors on developing autonomous cars helped pave the way for a safer and eco-friendly future. This new path is unavoidable and works hand-in-hand to humanity's destiny.

As early as 2014, Tesla Motors [9] had already implemented its Autopilot technology to its electric vehicles and the safety record it has achieved is astounding. The cameras and ultrasonic sensors have successfully predicted collisions with up to 76% accuracy and were able to prevent them with over 90% success rate.

Google, Tesla Motors and several other automobile companies conceive a future with reduced traffic problems, fewer road accidents (because humans errors) and more efficient public and private transport system starting in 2020. Already Tesla and Google have incorporated basic forms of autonomous vehicles capabilities into their cars (with the exception of Google only selling its LIDAR technology to car manufacturers and not

[9] https://www.tesla.com/

Figure 1.2.1: Tesla autopilot.

manufacturing their own cars as Tesla does). Tesla autopilot includes automatic braking, parking assistance, steering assistance, dynamic radar cruise control, and other features.

1.2.1 The Autonomous Vehicle Concept

The self-driving car initiative was first carried out decades before Google started doing technical research on the subject or even Tesla exists. The first recorded concept of an autonomous car was introduced in the 1939 New York World's Fair in the Futurama section. General Motors [10] (GM) created the exhibit as part of its vision of the future of America in 20 years time. Engineers and futurists included an automated highway system on which the self-driving cars would depend on to get people from one place to another.

But this idea imagined by General Motors took over 60 years before robotic vehicles started to be on the streets. However, they did not even need to create the automated highway system. Development full-fledged autonomous vehicles is gaining traction with the goal of making driving efficient and safe.

By 1958 - almost two decades since they introduced the concept during the World's Fair in New York. The self-driving car Norman Bel Geddes' (Figure 1.2.2) created for General Motors was finally realized. It relied on magnetized metal spikes embedded in the roadway and was remotely controlled by a device that guided the car by changing the electromagnetic fields in the spikes to keep the car inside its designated lane.

In 1977 the Tsukuba Mechanical Engineering Lab made some improvements to GM's self-driving car by using cameras linked to a computer, which could guide the car through the road at 32 km/h via image data processing. This was the first car to use image data as a layer for the autonomous task.

[10] https://www.gm.com/

Figure 1.2.2: The self-driving car Norman Bel Geddes.

Ten years later, two of Germany's leading car manufacturers, Daimler [11] and Mercedes Benz [12], collaborated on a project called VaMoRs. The VaMoRs was a 5-ton Mercedes Benz van equipped with cameras and other sensors These sensors send the data to an on-board computer that drive the car without human help, allowing the van to drive at 90 km/h on any road or highway and not collide or crash with other vehicles or objects.

The improvement of the technologies used in self-driving vehicles is directly proportional to how these types of vehicles perform on the road. Better tech means better autonomous cars, but developers were just starting to scratch the surface.

1.2.1.1 You Can't Talk About Self-Driving Cars Without Mentioning Elon Musk

The first car company that Elon Musk started was Tesla Motors. He intended Tesla to be an all-electric, clean and energy-efficient vehicle manufacturer to kick-start his dream of a green future and this future to Elon is electric. The company was incorporated in 2003 and Musk immediately held a press conference stating that he would develop affordable AEVs (autonomous electric vehicles) for the US and the rest of the world in 3 - 5 years time.

The first AEV coupe that Tesla Motors developed was the Tesla Roadster. It can go as far as 1000 km in a single charge and for an AEV released in 2012 with this specs was truly impressive! However, the Roadster cost 200,000 and obviously was not feasible for the masses, but even the subsequent Model S still cost twice as much as the 30,000 Tesla

[11] https://www.daimler.com/en/
[12] https://www.mercedes-benz.com/en/

sedan Musk promised. The first car that Tesla launched as an affordable car, at least on the US, was the Tesla Model 3.

From 2014 onward Tesla started to include the hardware needed for full self-driving capabilities in all of their vehicles even before the software/data was available, and yet Tesla had the best safety record compared to other auto brands in its class. Plus, Tesla's Autopilot feature in their vehicles has 360-degree coverage! Electronic sensors and cameras literally surrounding the car on every corner and can recognize cars and pedestrians on the road in various distances. This strategy allows the customer to truly believe in their company because with all the hardware it is only needed an update on his autopilot system which can be done by WIFI and they will have the last autopilot on the market.

The narrow forward camera has a maximum viewing distance of 250 meters allowing the car to have a view of which is going on and it is backed up by the secondary camera/sensor, the main forward camera, which can see up to 150 meters. The wide forward camera scans the peripheral vision of the car up to 200 degrees even though it can only see up to 60 meters out into the road. Finally, the rearward looking cameras cover over 180 degrees of the rear of the vehicle and can see up to 100 meters away. With all of these cameras the car has a complete view of the environment surrounded it.

All these cameras and sensors combine to form a fusion of sensors and eyes of the vehicle making Tesla cars almost military-grade, unmanned navigation systems. But as advanced as this technology may seem, Musk said that it is not at full self-driving capability yet. To full fill this task time is needed and more data to achieve it. Also there is an important part that has to be complete which concern to the governments of the different countries.

In January of 2019 Musk tweeted that in just 3 - 6 months Tesla motors would slowly depart from the autopilot feature and introduce its first-ever self-driving technology. As early as the year's end, Tesla car owners would be able to safely drive from Los Angeles to New York City without ever touching the vehicle's steering wheel. That's almost 4000 kilometers! I will be amazing to see if Musk can deliver on his promise.

But it's not just Tesla Motors that's making such promises to the world, in fact, about half of all the car manufacturers in the world intend to produce autonomous vehicles starting 2020 and onwards. So we might see a significant boost in intelligent transportation in the next decade or so which will be lead by these incredible companies.

1.2.2 Benefits of Self-Driving Vehicles

Just about every country has a government agency that deals with road safety and licensing drivers as well as the registration of vehicles. In the United States, the Department of Motor Vehicles (DMV) manages the issuance of driver's licenses, registering vehicles,

giving out unique plate numbers for traffic monitoring and police surveillance, as well as managing traffic and general road safety for drivers and pedestrians alike. Yet even under the strictest of traffic rules and regulations, there are still over 6 million car accidents that happen in America every year with around 3 million people who suffer injuries related to these accidents, and over 35.000 people get killed from them on average by the NHTSA.

This is where self-driving cars come in handy because scientists predict that they will significantly reduce, if not eradicate, car accidents in the near future.

Here are some of the benefits we can reap from incorporation autonomous vehicles into our everyday lives:

- Fewer accidents - computers do not feel fatigued, get hungry, nor get distracted while on the road. In fact, they only exist to drive the car and keep both passengers and pedestrians safe. Their cameras and sensors (which will be greatly improved in the future) are state-of-the-art and can detect and track people and vehicles from every angle of the car. They will be able to detect possible collisions and steer the vehicle safely to avoid accidents. It is estimated that accidents will be reduced by 5% per year in the first decade of implementation alone and could go down as much as 0.15% by 2040. This is an important improvement of autonomous vehicles because they can improve our life expectancy because humans will not die by car accidents.

- Decrease or totally eliminate traffic congestion the future of autonomous vehicles is to have them communicate with each other, thereby reducing traffic congestion and car accidents even more! Driving to your office 10 - 15 kilometres away will take less than half the time it before self-driving cars were allowed in the streets. Because they will be communicated and a computer has less latency than a human that allows the car to react earlier.

- Increase highway capacity - a Columbia University study [4] showed that if there were 100% self-driving cars occupying one lane of a given highway, then vehicle capacity could be increased to as much as 12,000 cars per lane per hour driving at a constant speed of 120 km/h. None of our current traffic technology could achieve this feat despite government expenditures reaching several hundred million dollars per year.

- Lower fuel consumption - assuming that all self-driving cars will be hybrid or all-electric. If electricity was also harvested from clean and renewable energy sources, then that would reduce fuel consumption even further.

- Enhanced human productivity - riding a bus, train or airplane gives passengers more time to do work and other important things versus driving their own car. Now

imagine if half of all vehicles were replaced with self-driving cars? Productivity levels would sharply increase and passengers would be able to accomplish more in the same amount of time. Also humans not only could work during their trips to work they will also complete it in less time improving their life style.

- Car parking space problems will be eliminated - self-driving cars would also be able to communicate with parking lots to immediately locate good parking. No more hunting for parking space like the old days which takes at least 5 minutes of circling and wastes precious time which can be increased to near an hour in city centers.

- Improved mobility for children, elderly, and disabled - there are three types of people who will have trouble driving cars because of their age or problems that do not allow them to drive. Some of these folks completely have to rely on themselves as they no longer have any relatives or next of kin. Giving them the option to own self-driving cars will greatly increase their self-reliance and enable them to function more without needing further assistance.

- Elimination of traffic enforcement personnel - human traffic enforcers make errors and statistically make traffic congestion even worse. Self-driving cars will eliminate the need for them, making them available for other good services to society.

- Higher speed limits - because self-driving cars will be able to communicate with each other, predict collisions, and adjust the car's speed and direction faster than humans, cars will be able to travel faster. Faster travel times could mean never being late for work, school or an important meeting ever again.

- Self-driving cars have been in the making for a long time. The good news is that they are finally here and on the road. Undoubtedly, they will make our transportation safer and more efficient. It is just a matter of time before we are all able to text and ride and leave the driving to the car itself.

1.3 State of the art

In last years, important advances have been made in autonomous driving field from an academic and industrial point of view. According to SAE (J3016), five Levels of Automation can be applied, achieving the full automation in the Level 5. Nowadays, we have technologies that can be used to take over the functions normally reserved for the driver. In Level 1 and Level 2 the driver is still supposed to be fully engaged in supervising the actions of the vehicle under all circumstances. In Level 3 and Level 4, the driver is freed from supervision, either in limited situations or during the entire trip, as mention early. Problems arise at the above levels, because the driver is in-the-loop and not always aware

of what is happening (Level 1 and Level 2) or is out of the loop and needs quickly to be brought back into the loop for some unexpected reason (Level 3 and Level 4) [5].

In this context of shared control between human and machine, driving in a safety way requires ensured that the state of driver is suitable for driving. This is particulary important when the vehicle requests the driver intervenes in case the driving scenario is complex. In addition, it is crucial to assess driver'awareness of driving scenario (e.g. surronding vehicles or pedestrian) right before the take-over. In conclusion, evaluating driver's visual attention is a key task in the development of automated vehicles. Gaze tracking estimation is the common way for evaluating the driver's visual attention [6].

In the literature, there are different approaches mainly based on head-mounted eye trackers [7], [8], [9] (Figure 1.3.1), active desktop-mounted eye trackers [10] (Figure 1.3.2) and passive desktop-mounted cameras [6], [11], [12], [13], [14]. The former provide accurate gaze information but they are intrusive and costly. The latter are low-cost and non-intrusive but less accurate. Most of the existing visual attention systems have been validated in simulation with simple scenes. However, the simulation of driver attention in complex driving scenarios is rather challenging and highly subjective [2]. The goal of these systems is to automatically estimate the regions of interest where the driver is looking. In this sense, acquisition systems and computer vision techniques have gained importance regarding the traditional human factor approaches focused on driving behavior understanding using manual tools.

Table 1.1 shows some eye-trackers available on the market and their comparison among them with the one proposed on this work.

Figure 1.3.1: EyeLink II, head-mounted eye trackers.

There are different challenging databases to validate driver visual attention in the state

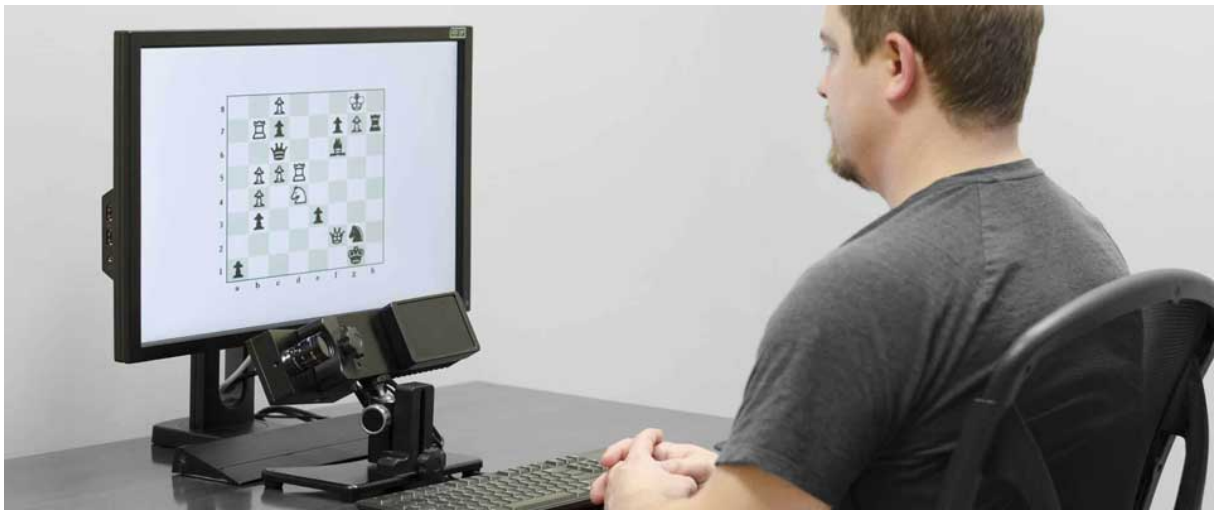


Figure 1.3.2: EyeLink 1000 Plus, desktop-mounted eye tracker.

Figure 1.3.3: Tobii Pro Glasses 3, glasses eye tracker.

of the art. Recently, the project DR(eye)VE [15] collected 555,000 frames with driver attention maps. However, only few critical scenes were captured for only one driver. Berkeley DeepDrive Attention (BDDA) dataset [10] includes 1232 videos with critical situations in-lab designed. However, the critical situations do not cause true accidents and in consequence the attention simulation does not explore the dynamic process from critical situation to actual accidents. DADA2000 dataset [2] is a larger and more diverse video benchmark with driver attention and driving accident annotation simultaneously. It contains 2000 video clips with fairly complex accidental scenarios in diverse weather and lighting conditions. Table 1.2 shows the comparison of different driver attention datsets.

In this context, there have been some different Advanced Driver Assistance Systems (ADAS) that have improve the security on the road. But the systems more extended in actual vehicles that anyone can find on the market are related to monitoring systems

Table 1.1: Eye-tracker comparison.

Eye_tracker	Kind	Vehicle use	Accuracy (degree)	Price
EyeLink II	Head-mounted	No	0.5	20.000,00 €
EyeLink 1000 Plus	Desktop-mounted	No	0.25-0.50	26.000,00 €
Tobii Pro Glasses 3	Glasses	Yes	0.1	32.000,00 €
SMI RED250	Desktop-mounted	No	0.4	41.000,00 €
Ours	Camera	Yes	0.56	500,00 €

Table 1.2: Attribute comparison of different driver attention datsets.

Dataset	Rides	Duration (hours)	Drivers	Gaze providers	Event	Gaze patterns for each frame
Dr(eye)VE [15]	74	6	8	8	464 braking events	Attention map of a single person
BDDA [10]	1232	3.5	1232	45	1427 braking events	Average map of multiple observers
DADA2000 [2]	2000	6.1	2000	20	2000 accidents (54 categories)	Raw attention maps of multiple observers

that assist the driver when he is not doing a correctly driving behaviour but without understanding the human behaviour just being focus on the car state. Suffering from the complexity of capturing driver attention. Then one of the key factor in road fatalities is the absence of driver attention as demonstrated in recent studies as [16].

Other works [15], [17], [18] and [19] are trying to solve the problem with computer vision model that try to replicate where drivers look in complex driving situation like accidents or braking events, but to achieve this task it is necessary a huge amount of data that nowadays is not enough [20], [21] and [2]. All together being composed by 15,6 hours of video. Which contribute to achieve this models but it is still needed more data to get the accuracy required in security vehicle systems.

These dataset acquire driver's gaze from an accurate eye tracking device which restrict the contributions from the community because of the the price of this device.

Also the difficult to replicate the scenarios in real life conditions complicate the obtaining of these models because there is more hard to find patterns between different frames that are not evaluate by more than one person. To being able to can replicate scenarios, we understand that in real driving conditions is such a difficult way of work because you can not make the scene conditional to some aspects that you want to replicate.

So in order to continue with this task of acquiring data we proposed the use of Open-

Face [1], because as probed on this work we think is enough accurate to get driver gaze. Also is a tool that can be easily implemented in cars to get more data because of the small price (normal camera can be used) and the way to acquire data without disturbing the driver or changing his behaviour cause of framework implemented on his vehicle.

The way humans understand a driving scene is essential to improve autonomous driving, also this can be to warn the driver when he is not doing a responsible driving behaviour (drowsiness, using mobile phone, etc.).

Attention maps that are not built using eye-trackers are usually made with two main computational models: top-down and botton-up. But we think these are not the best way to get a Ground Truth for a dataset because are based on models not in real human attention.

To get gaze estimation there are some tools and not open source systems that require specific hardware such as head mounted cameras or infrared, that to authors opinion are more intrusive to the driver, [7], [8], [9]. There also exist some commercial systems that use a camera such as xLabs and EyesDecide.

Other works as [22], [23], [24] and [12] use similar techniques as that we proposed but they do not have being tested against infrared eye tracker models that are the best way to get driving attention but with the problem of to have some glasses that differs from what we think is a Naturalistic driving studio, because the driver does not drive as he is used to.

Chapter 2

Tech4AgeCar

2.1 Introduction

Statistics show that 68% of the population in the European Union (EU), including associated states, is living in urban areas. According to the World Health Organization, nearly one third of the world population will live in cities by 2030. Aware of this problem, the Transport White Paper published by the European Commission in 2011 indicated that new forms of mobility should be proposed to provide sustainable solutions for people and goods safely. Regarding safety, it sets the ambitious goal of halving the overall number of road deaths in the EU between 2010-2020. However, the goal will not be easy, only in 2014 more than 25,700 people died on the roads in the EU (18% reduction). Besides, some studies show that fatal accidents increase with age for people 65 years and older.

Autonomous driving is considered as one of the solutions to the before mentioned problems and one of the great challenges of the automotive industry today. The existence of reliable and economically affordable autonomous vehicles will create a huge impact on society affecting environmental, demographic, social and economic aspects. In particular, it is estimated to cause a reduction in road deaths, improved traffic flow, reduced fuel consumption and harmful emissions associated, as well as an improvement in the overall driver comfort and mobility in groups with impaired faculties, like the elderly or disabled people.

Autonomous driving has attracted much attention recently by the research groups and industry, due to the billboards of various companies on expectations of market entry. However, his predictions seem to be very optimistic. A more scientific organization, such as IEEE, has recently predicted that by 2040 the majority of vehicles traveling on highways will be autonomous. Driving in urban environments will take longer, due to its complexity and uncertainty.

With this background, this project will focus on the research in technologies for the

development of an autonomous electric car to assist the elderly population in predominantly urban environments. The proposal is disruptive because it raises new techniques applied on a future electric car, where Spain is an international reference, and targeted to a sector of the population with growing needs. The proposed system is based on the most advanced techniques of sensory perception and in-vehicle mapping, control and path planning in dynamic and changing environments.

A sensor fusion architecture based primarily on computer vision, laser and GPS technologies will be developed. The architecture will allow the mapping of the environment, the semantic classification of the scene and the real-time detection of obstacles in the path. This, together with vehicle positioning technology based on digital maps, will establish the planning and control algorithms to carry out the independent movement of the vehicle from a source to a destination provided by the user. Additionally, a study of the behavior of senior drivers will take place, to identify shortcomings and limitations, and develop a customized HMI (Human Machine Interface).

2.2 Vehicle

In 2017 Robesafe group bought an open source chassis TABBY EVO developed by Open Motors [1]. This platform is open source, which means that its blueprints and electric diagrams are available to check by owners, and can be changed by developers without any problem. This allows the developers to implement new systems to the car and build a better approach to improve the car industry and specially on the electric manufacturer cars.

Figure 2.2.1 shows the evolution that the car has experimented in these years in the Robesafe group. Firstly it was just the chassis, four seats and the motor, after that, a tubular structure was built to place the sensor (LIDAR, stereo camera, DGPS, etc.) and currently there is a car concept which is completely prepare to an autonomous driving.

2.3 Vehicle layers

With a platform to implement the required systems to build an autonomous vehicle, it was time to begin the developing of the different architectures/layers that would control the vehicle, as can be seen in Figure 2.3.1

[1] https://www.openmotors.co/

Figure 2.2.1: Tech4AgeCar evolution.

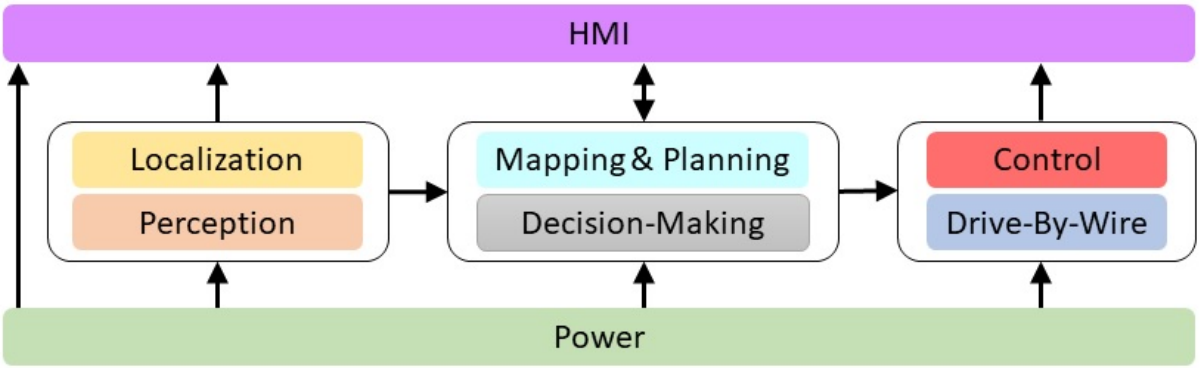


Figure 2.3.1: Tech4AgeCar architecture.

2.3.1 Drive by Wire layer

The low control level of the vehicle and one of the most important system of it because without this layer, all the others layers will not work.

In this layer the power supply with all the batteries that power the vehicle has been developed, the steering wheel has been changed to be controlled by an electric signal, as the throttle pedal. Also a switching control between automatic and manual has been developed to change between both modes in a safety way giving the steering wheel, throttle and brake pedal control to the driver as soon as he requires it.

2.3.2 Localization layer

This layer is composed by two main systems that work together to give the best possible results, the result expected is the position of the car on the map. This is a very important task because the vehicle has to be drive with this input.

The position of the car is obtained by a GNSS (Global Navigation Satellite System) and odometry based on two encoder placed on the rear wheels. The odometry estimates the vehicle position, measuring the movement of the rear wheels with two encoders. These encoders have 360 pulses per turn. The GNSS (Global Navigation Satellite System) are the systems GPS, Glonas y Galileo. To have a better accuracy differential GNNS are used (DGNSS) and RTK (Real Time Kinematic) [25]. Both systems are fused with an EKF (Extended Filter Kalman) as shows Figure 2.3.2.

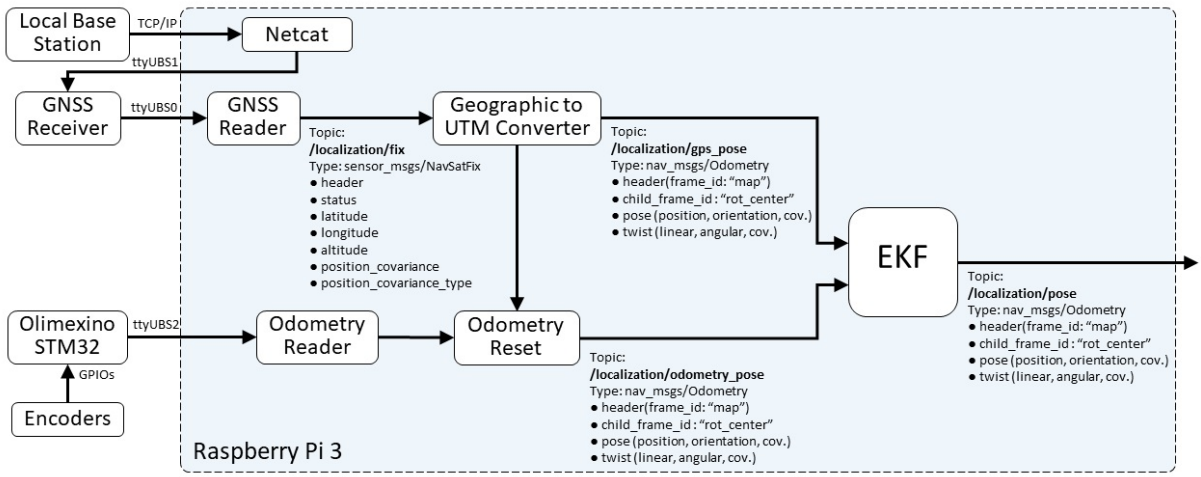


Figure 2.3.2: Localization module.

2.3.3 Control layer

Control layer is the one in charge to lead the vehicle when it is in autonomous mode, to achieve this task. It has to work with the other layers in order to have an optimum control. This layer generates local paths to get to the waypoints generated by the Mapping Planning layer.

To follow the paths the the Pure Pursuit algorithm [26] is used that searches for the next waypoint in the parh. Also this is in charge of the obstacle avoidance, which use an adaptation of the BCM (Beam Curvature Method) [27].

2.3.4 Mapping Planning layer

This layer is in charge of the generation of the waypoints to complete the path. To achieve this task Lanelets [28] are used. To make this task possible a lanelet map of the University of Alcalá Campus has been built using the tool JOSM.

Once the map and the path is set, the waypoints are calculated using A* algorithm [29]. This waypoints are the path planning that the control layer has to follow in order to achieve the goal destiny.

2.3.5 Decision making layer

This layer depends of Mapping Planning and Perception layer, because is the one in charge of taking high lead decisions on the road. This layer is lead by a Petri net which has been programmed to take information from the environment through the map and the perception and take decisions based on them. Some of the decisions that can take the car are: overtaking, ACC (Adaptive Cruise Control), lane keeping, etc.

2.3.6 Perception layer

This is a challenging part of the project that is the one that understand the environment using sensors. For this purpose two sensors are used, LIDAR and RGBs cameras. These sensors have to detect static objects as road, trees, traffic signals, etc; and dynamics objects as vehicles, pedestrian, etc.

For this purpose Neuronal Networks are used. Two of them are used from a RGB image. One is in charge of detection through a YOLOv3 [30] and another one is in charge of generating a semantic segmentation through a ERFNet [31]. With this information and the point cloud generated by the LIDAR, a 3D point cloud coloring is built to understand the environment, as show on Figure 2.3.3.

2.3.7 Vehicle to User layer

This layer is the main goal of this work and will be detailed in the following chapters. But as an introduction, this is the layer in charge of showing information from the autonomous system and taking orders by the user. It works as a bridge between the car and the driver.

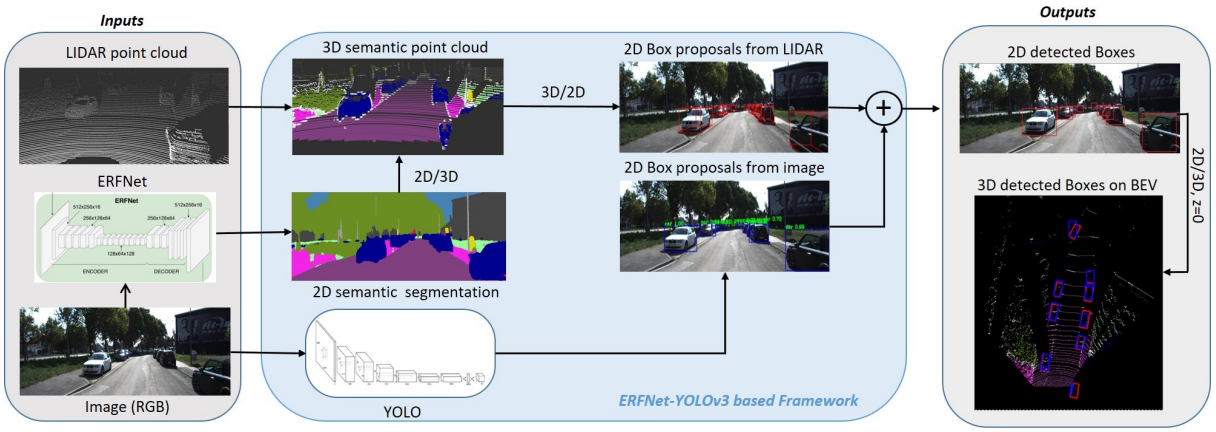


Figure 2.3.3: Perception layer.

Chapter 3

Theoretical study

3.1 Introduction

In this chapter we introduce the different techniques used during this work in order to achieve the proposed task.

This chapter is theoretical because in the next chapters the explained algorithms will be explained as they have been implemented on this Master Thesis.

3.2 Techniques used

3.2.1 NARMAX

The Nonlinear AutoRegressive Moving Average model with eXogenous inputs (NARMAX model) [32] represents a wide class of nonlinear systems, which means that the signal modeled depends of:

- Past signal terms (AR)
- Past noise terms (MA)
- Other signal with a possible delay (X)

These relationships among them can be non linear and the modeled signal is discrete, which can be defined as:

$$\begin{aligned} y(k) = F[&y(k-1), y(k-2), ..., y(k-n_y), \\ &u(k-d), u(k-d-1), ..., u(k-d-n_u), \\ &e(k-1), e(k-2), ..., e(k-n_e)] + e(k) \end{aligned} \tag{3.1}$$

Where $y(k)$, $u(k)$ and $e(k)$ are the system output, input, and noise sequences respectively; n_y, n_u, and n_e are the maximum lags for the system output, input and noise; $F[]$ is some unknown nonlinear function, d is a time delay typically set to $d = 1$.

NARMAX is based on five steps that answer the following questions:

1. Structure detection - "What parts are in the model?"
2. Parameter estimation - "What are the values of the model coefficients?"
3. Model validation - "Is the model correct and unencumbered?"
4. Prediction - "What does the modeled signal look like in the future?"

There are some methods that achieve these steps such as FROLS, MFROLS. These algorithms try to find the most simple model that describes the structure of the the system.

The problem is that the developer has to suggest possible types of non linearity because the number of non linearity is infinite. These can be:

- Polynomial degree
- Term degree
- Logarithm
- Neuronal network
- Superposition of the above methods

For this reason, modeling a non linear system brings the possibility of unknowingly making a fatal error. Because if the system under test has a non linearity that was not assumed, the modeling result cannot be corrected. In NARMAX methods, the only limit is the assumptions made by the modeler. Here after some non linear modelling systems are presented.

3.2.1.1 OLS

This work will propose to use the Orthogonal Least Squares (OLS) algorithm to establish the correspondence between the face features based on the camera and the coordinates and the eye gaze mapping on a screen. This approach has been used in non linear systems because OLS will search through all possible candidate model to select the better approach.

The main idea of the OLS estimator is to introduce an auxiliary model whose elements are orthogonal to the signal that is modeled, as in the Wiener Series. Then, subsequent parts of the orthogonal model can be determined in turn, and then the parameters of the searched model can be calculated based on them. Iterative repetition of the steps of this method allows not only to find unbiased model estimators, but also to show what contribution each of them had to the final modeling result.

Considerations should begin with the assumption of the general model:

$$y(k) = \sum_{i=1}^{M} \theta_i p_i(k) + e(k) \tag{3.2}$$

Where $y(k)$ is the modelled system response for $k = 1, 2, ..., N$; θ are the model parameters ($\theta \in R$) located at the regressors $p_i(k)$; $e(k)$ is the external noise or error at the moment $k = 1, 2, ..., M$.

Regressors $p_i(k)$ are defined as the combination of delayed signal values or delayed external signals. In the general case, the function of the delayed members of the model can take any non-linear form. For the model, it is also assumed that each regressor $p_i(k)$ is independent of the model parameters θ, therefore:

$$\frac{\partial p_i}{\partial \theta_j} = 0 \text{ for } i = 1, 2, ..., M \text{ and } j = 1, 2, ..., M \tag{3.3}$$

The goal of the estimator is to transform the model specified in equation 3.2 into an auxiliary model whose elements are orthogonal to each other. This type of model has the form:

$$y(k) = \sum_{i=1}^{M} g_i(k) q_i(k) + e(k) \tag{3.4}$$

Where g_i are the parameters of the orthogonal model; $q_i(k)(i = 1, 2, ...M)$ are the orthogonal components of the model. The orthogonality condition is then presented as:

$$\sum_{k=1}^{M} g_i(k) q_i(k) = \begin{cases} d_i = \sum_{k=1}^{M} q_i^2(k) \neq 0, & \text{i = j} \\ 0, & \text{i} \neq \text{j} \end{cases} \tag{3.5}$$

The orthogonalization procedure for the model from equation 3.2 can be summarized as:

$$\begin{cases} q_1(k) & = p_1(k) \\ q_2(k) & = p_2(k) - a_{1,2}p_1(k) \\ q_3(k) & = p_3(k) - a_{1,3}p_1(k) - a_{2,3}p_2(k) \\ & \vdots \\ q_m(k) & = p_m(k) - \sum_{r=1}^{m-1} a_{r,m}q_r(k),\ m = 2,3,...,M \end{cases} \tag{3.6}$$

Where parameter $a_{r,m}$ associates the components of the output model from equation 3.3 with the orthogonalized model according to the following form:

$$a_{r,m} = \frac{\sum_{k=1}^{N} p_m(k)q_r(k)}{\sum_{k=1}^{N} q_r^2(k)},\ 1 \leq \text{r} \leq \text{m-1} \tag{3.7}$$

On the base:

$$g_i = \frac{\sum_{k=1}^{N} y(k)q_i(k)}{\sum_{k=1}^{N} q_i^2(k)},\ \text{i} = 1,2,...,\text{M} \tag{3.8}$$

Therefore:

$$\begin{cases} \theta_M & = g_M \\ \theta_{M-1} & = g_{M-1} - a_{M-1,M}\theta_M \\ \theta_{M-2} & = g_{M-2} - a_{M-2,M-1}\theta_{M-1} - a_{M-2,M}\theta_M \\ & \vdots \\ \theta_m & = g_m - \sum_{j=m+1}^{M} a_{m,j}\theta_j,\ m = M-1, M-2; ..., 1 \end{cases} \tag{3.9}$$

This allows, based on previously established regressors, to develop a model consisting of orthogonal components, and to recreate the basic model from this type of model.

3.2.1.2 ERR

The problem in carrying out OLS estimation is the criterion for selecting subsequent regressors for the orthogonal model being created. Remembering that the model also has noise e, the energy (or variance) of the system can be represented as:

$$\frac{1}{N}y^T y = \frac{1}{N}\sum_{i=1}^{M} g_i^2 q_i^T q_i + \frac{1}{N}e^T e \tag{3.10}$$

Where y is the modelled signal vector; g_i are the coefficients standing next to elements of the orthogonalized model; q_i are the vector elements of the orthogonalized model; e is the vector of noise samples; N is the number of signal samples.

$$1 = \frac{\sum_{i=1}^{M} g_i^2 q_i^2 q_i}{y^T y} + \frac{e^T e}{y^T y} = \sum_{i=1}^{M} ERR_i + ESR \tag{3.11}$$

Where ERR_i is the Error Reduction Ratio and ESR is the Error to Signal Ratio.

To measure the significance of each model parameter the Error Reduction Ratio (ERR) is used, which indicates system improvement, in percentage (0-1), can be accounted by including the model parameters. This capability is important for the model in order to get the best model possible without getting a complex model. ERR allows the model to get only the best parameters in order to achieve best model performance without a high training time.

The sum of subsequent values ERR can mean the end of modelling when $\sum_{i=1}^{M} ERR_i \rightarrow 1$. Comparison of individual values ERR for various elements of the model shows which of them are the most influenced on the signal of the designated model.

A model was assumed as a linear combination of three signals and external noise. In order to complicate the modelling process, it was assumed that the model sought depends on four signals, where the additional fourth signal was determined as a linear combination of two component signals of this model and other noise. A desirable feature of a given estimator is to indicate the correct components of the model and to reject the signal that has no effect on it. The specified linear model has the form:

$$y(k) = x_1(k) + x_2(k) + x_3(k) + e(k) \tag{3.12}$$

Where $x_1(k)$, $x_2(k)$ and $x_3(k)$ are certain signals and $e(k)$ is the noise.

An additional signal is introduced, $x_4(k)$, which is a linear combination of the two other signals making up the model and the noise $\eta(k)$.

$$x_4(k) = 0,25x_1(k) + 0,75x_2(k) + \eta(k) \tag{3.13}$$

Therefore, the model sought is:

$$y(k) = \theta_1 x_1(k) + \theta_2 x_2(k) + \theta_3 x_3(k) + \theta_4 x_4(k) + e(k) \tag{3.14}$$

Which will be calculated following the next algorithms.

3.2.1.3 FROLS

Using the OLS estimator and coefficient ERR, describes the FROLS (Forward Regression with Orthogonal Least Squares) nonlinear modelling algorithm, a forward regression algorithm using the orthogonal sum of the least squares. It was assumed that the selection of subsequent regressors for the OLS estimator should be conditioned by the highest value ERR for a given regressor. This allows choosing the model member that reduces the modelling error to the greatest extent and to stop modelling when ESR will have a satisfactory value. In the following sections, steps in modelling according to the FROLS algorithm will be presented.

3.2.1.3.1 Step 1. Data collection In order to perform modelling, acquisition of the test signal waveforms is required, $y(k)$, as well as external input signals, $x(k)$, which affect the system under study. The signal should be collected as discrete or transformed into such a form. In addition, the recorded waveform should have as little external noise as possible and not be subjected to filtration, which may disturb the identification process. According to the author of the method, if the noise presents in the signal is zero-mean white noise, it does not affect the ERR value and the identification process.

3.2.1.3.2 Step 2. Defining the modelling framework Because the number of possible nonlinearities is infinite, a determination a priori the area is required in which the search is being made. These requirements can be formulated as the following questions:

- What is the maximum delay of the AR term (n_y)? (Output signal).
- What is the maximum delay of external signals (n_x)? (Input signal).
- What nonlinearities are predicted and what is their maximum degree (l)?

After the developer has provided the answer to these questions, the following parts of the model are known: $y(k-1)$, $y(k-2)$, ..., $y(k-n_y)$, $u_i(k-1)$, $u_i(k-2)$, ..., $u_i(k-n_{x_i})$. Where y is the signal under testing, u_i one of the external input signals, n_y the maximum delay of the signal under test and n_{x_i} the maximum delay of one of the external input signals. The non-linearity that binds these members is also known, which allows the next step to be taken. In the case where the non-linearity is a polynomial, the maximum degree of non-linearity determines the maximum power occurring in this polynomial.

3.2.1.3.3 Step 3. Determination of the regressor vector Knowing the modeling framework defined in the previous step, you must determine a vector containing all possible regressors for a given signal. For example, the searched model has an output y

and one input u with $n_y = n_x = 2$ and that nonlinearity is a polynomial that is up to the second degree maximum, $l = 2$. Then, the regressor vector will consist of a combination of signals: $y(k-1)$, $y(k-2)$, $u_i(k-1)$, $u_i(k-2)$ according to the maximum polynomial degree. Including the constant component marked as *const.*, regressor vector marked as D, will be defined as:

$$\begin{aligned} D = \{const., y(k-1), y(k-2), u(k-1), u(k-2)\} \cup \\ \{y^2(k-1), y(k-1)y(k-2), y(k-1)u(k-1), y(k-1)u(k-2)\} \cup \\ \{y^2(k-2), y(k-2)u(k-1), y(k-2)u(k-2)\} \cup \\ \{u^2(k-1), u(k-1)u(k-2)\} \cup \\ \{u^2(k-2)\} \end{aligned} \tag{3.15}$$

The number of regressors is determined by:

$$M = \binom{n+l}{l} = \frac{(n+l)(n+l-1)...(n+1)}{l!} \tag{3.16}$$

Where $n = n_y + n_{x_1} + ... + n_{x_i}, l$ is the maximum degree of the polynomial.

3.2.1.3.4 Step 4. Choosing the first element Once the regressor vector has been specified, the database of the crated model is known. Choosing the first one requires determining the value of ERR for each element of the vector $D = p_1, p_2, ..., p_m$ where m is the number of potential regressors.

$$g_m = \frac{y^T p_m}{p_m^T p_m} \tag{3.17}$$

$$ERR[m] = \frac{g_m^2 p_m^T p_m}{\sigma} \tag{3.18}$$

$$l_1 = arg \max_{1 \leq m \leq M} ERR[m] \tag{3.19}$$

Where l_1 corresponds to the highest regressor and it is accepted as the first element of the orthogonal model. The parameter g and ERR is saved.

$$q_1 = p_{l_1} \ ; \ g_1 = g_{l_1} \ ; \ err[1] = ERR[l_1] \ ; \ D_1 = D - p_{l_1} \tag{3.20}$$

After saving the found regressor and its associated values, the search is restarted to find the next member of the orthogonal model.

3.2.1.3.5 Step 5. Selecting the next elements of the model The next steps are performed analogously to the first step, with the difference that the orthogonal model already has a certain number of terms, depending on the number of steps previously performed. Value Calculation ERR for potential new members of the model must be made again, in relation to the current form of the model. Therefore, the regressor index is searched (marked as l_s) about the highest ERR for the current model form, where s) is the current step number. The regressor vector is marked as D_{s-1} and does not contain the members selected in the previous steps. Then, for $m = 1, 2, \ldots, M - (s-1)$ and $m \neq l_1 \neq l_2 \neq \ldots \neq l_{s-1}$ we have:

$$q_m = p_m - \sum_{r=1}^{s-1} \frac{p_m^T q_r}{q_r^T q_r} q_r, p_m \in D_{s-1} \tag{3.21}$$

$$g_m = \frac{y^T q_m}{q_m^T q_m} \tag{3.22}$$

$$ERR[m] = \frac{g_m^2 q_m^T q_m}{\sigma} \tag{3.23}$$

$$l_s = arg \max_{1 \leq m \leq M-(s+1)} ERR[m] \tag{3.24}$$

After finding l_s, which corresponds to the highest regressor ERR, the orthogonal model is extended by another member, q_s, along with the corresponding parameter g_s. The ERR value for the selected element is saved, and from the vector D the selected regressor is deleted. In addition, parameter values are determined $a_{r,s}$ according to the following equations:

$$q_s = q_{l_s} \ ; \ g_s = g_{l_s} \ ; \ err[s] = ERR[l_s] \ ; \ D_s = D_{s-1} - p_{l_s} \tag{3.25}$$

$$a_{r,s} = \frac{q_r^T p_{l_s}}{q_r^T q_r}, r = 1, 2, \ldots, s-1 \tag{3.26}$$

This step is repeated until the ESR value is satisfactory:

$$ESR = 1 - \sum_{s=1}^{M_0} err[s] \tag{3.27}$$

Where M_0 is the number of selected regressors. A certain value limit is necessary to enter ESR which lets you decide the end of the modelling.

3.2.1.4 Modelling process analysis

An important decision when performing modelling with the NARMAX method is when to stop adding more regressors to the model. The right moment does not have a strict definition, but rather is determined by a set of factors whose superposition is a premise to take it. In the case of the NARMAX method, one of the basic parameters is the ERR sum for the model, called $SERR$:

$$SERR = \sum ERR \tag{3.28}$$

The two main parameters on the basis of which the current model is evaluated are: $SERR$ value and its course during modelling. It is assumed that a well-fitted model has $SERR$ higher than 0.8 [33].

3.2.1.5 Determination of the final model

When M_0 regressors are selected, it is necessary to determinate the parameters that are next to them. Assuming a low value of ESR, the general and ortogonalized models are equal to:

$$y(k) = \sum_{i=1}^{M_0} \theta_i p_i(k) + e(k) = \sum_{i=1}^{M_0} g_i q_i(k) + e(k) \tag{3.29}$$

In the above equation, the only unknowns parameters are, θ_i, because regressors, g_i, were selected during modelling, and the elements of the orthogonal model $q_i(k)$ along with their parameters were appointed on an ongoing basis during the process. Therefore, it is necessary to perform the conversion from the orthogonal model to the initial model by performing an inverse process to the previously performed orthogonalization. Knowing the values of the elements $a_{r,s}$ for $1 \leq r < s \leq M_0$ the following matrix can be specified:

$$A = \begin{pmatrix} 1 & a_{12} & a_{13} & \dots & a_{1M_0} \\ 0 & 1 & a_{21} & \dots & a_{2M_0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & a_{M_0-1,M_0} \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix} \tag{3.30}$$

Which can be included in equation:

$$A\theta = g \tag{3.31}$$

Where $g = [g_1, g_2, \dots, g_{M_0}]$, $\theta = [\theta_1, \theta_2, \dots, \theta_{M_0}]$. Solving this equation, model parameters are obtained, thus determining the full output model.

3.2.2 Heat Map

For visualization a method called heat map is used which is composed by two Gaussian distribution, one for each axis. This method gives a color gradient in function of a probability color, Figure 3.2.1 represents the different color gradients used by this method. In this work, we use the option B that is composed by 5 main colors.

The option chosen have 5 main colors [blue, cyan, green, yellow and red], which goes from a probability of 0 to 1. This allows the user to clearly see which point has more probability to be the watched point.

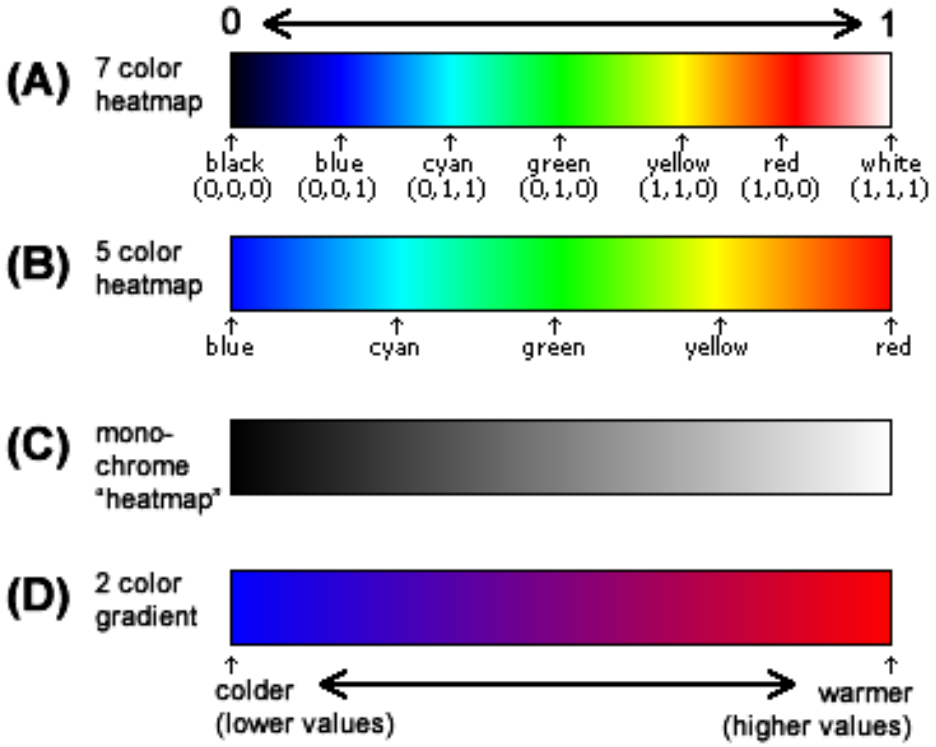


Figure 3.2.1: Colour gradients of a heat map

In order to get the probability of the function, a Gaussian method is used:

$$p(x) = e^{-\frac{1}{2}(\frac{x-\mu_x}{\sigma_x})^2} \tag{3.32}$$

Where x is the sample to measure the probability, μ_x is the mean of the temporal window used and σ_y is the standard deviation of the temporal window on the x axis. This calculus is done twice in order to get the probability of the sample in the x axis and in the y axis.

$$p(y) = e^{-\frac{1}{2}(\frac{y-\mu_y}{\sigma_y})^2} \tag{3.33}$$

The final probability which will give us the color is done like:

$$p = p(x)p(y) \tag{3.34}$$

Figure 3.2.2 shows how the heat map is done with the two Gaussians probability for each axis. Both probability are multiplies and gives the final probability that will be transform to colors in order to represent it.

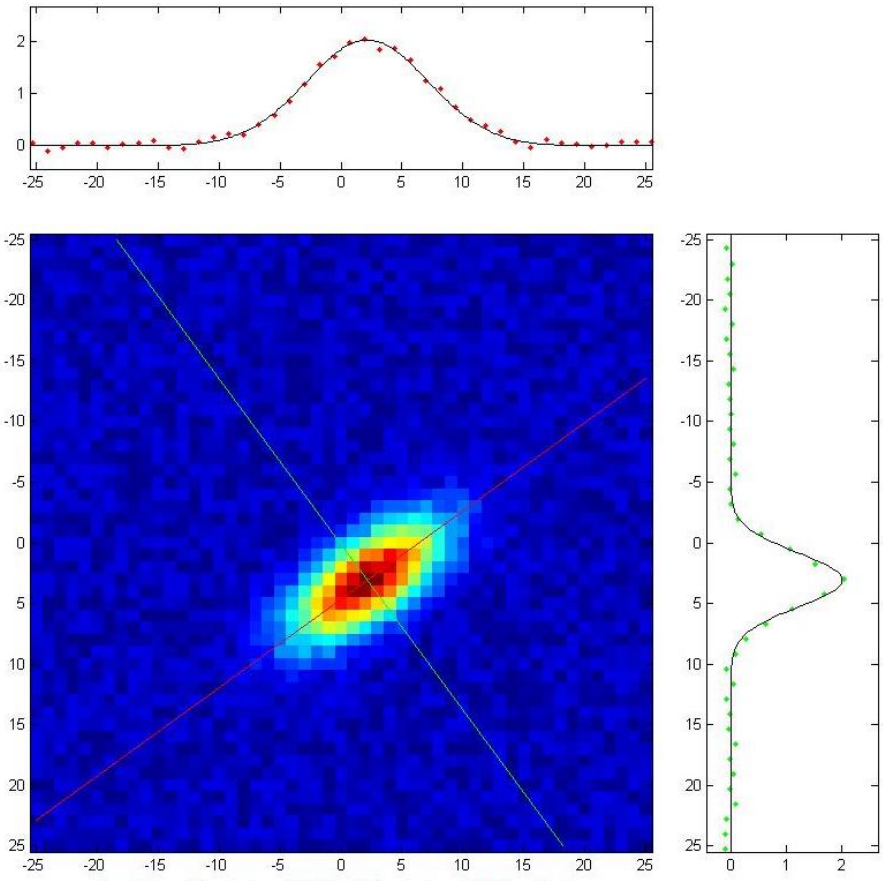


Figure 3.2.2: Heat map generation

Chapter 4

Software and technologies used in this thesis

4.1 Introduction

This chapter introduces the software and different technologies implemented on this work and different tools and models used to achieve the proposed task.

4.2 Robot Operating System

The Robot Operating System (ROS) [34] is a flexible framework for coding robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms.

Figure 4.2.1: Robot Operating System (ROS).

ROS was launched because creating truly robust, general-purpose robot software is hard. From the robot's perspective, problems that seem trivial to humans often vary wildly between instances of tasks and environments. Dealing with these variations is so hard that no single individual, laboratory, or institution can hope to do it on their own.

As a result, ROS was built from the ground up as a framework to encourage collaborative robotics software development. For example, one laboratory might have specialists in mapping indoor environments, and could contribute a world-class system for producing

maps. Another group might have experts at using maps to navigate, and yet another group might have discovered a computer vision approach that works recognizing small objects in clutter. ROS was specifically designed for groups like these to collaborate and build upon each other's work, as is described throughout this site.

ROS is a large project with many ancestors and contributors. The need for an open-ended collaboration framework was felt by many people in the robotics research community, and many projects have been created towards this goal.

Various efforts at Stanford University in the mid-2000s involving integrative, embodied AI, such as the Stanford AI Robot (STAIR) [35] and the Personal Robots (PR) program, created in-house prototypes of flexible and dynamic software systems intended for robotics use. In 2007, Willow Garage, a nearby visionary robotics incubator, provided significant resources to extend these concepts much further and create well-tested implementations. The effort was boosted by countless researchers who contributed their time and expertise to both the core ROS ideas and to its fundamental software packages. Throughout, the software was developed in the open using the permissive BSD open-source license, and gradually has become a widely-used platform in the robotics research community.

From the start, ROS was developed at multiple institutions and for multiple robots, including many institutions who received PR2 robots from Willow Garage. Although it would have been far simpler for all contributors to place their code on the same servers, over the years, the "federated" model has emerged as one of the great strengths of the ROS ecosystem. Any group can start their own ROS code repository on their own servers, maintaining full ownership and control of it. They don't need anyone's permission. If they choose to make their repository publicly available, they can receive the recognition and credit they deserve for their achievements, and benefit from specific technical feedback and improvements like all open source software projects.

The ROS ecosystem now consists of tens of thousands of worldwide users, working in domains ranging from tabletop hobby projects to large industrial automation systems.

4.3 Linux

The operative system used in this Master Thesis is Ubuntu [1], which is a free and open-source Linux distribution based on Debian. Two different releases have been used in order to install the ROS distribution that is able to be installed because of different ROS package are needed. The two releases used have been Ubuntu 16.04 (LTS) and Ubutnu 18.04 (LTS) which means that they will have long-term support.

[1] https://ubuntu.com/

4.4 Docker

Docker [36] is an open-source project that automates the deployment of applications within software containers, providing an additional layer of abstraction and automation of application virtualization across multiple operating systems.

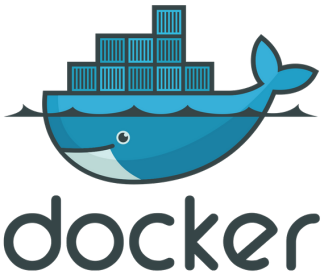


Figure 4.4.1: Docker.

Inside containers developers can accommodate all the dependencies that the application requires to be executed, as the source code, the system libraries, the framework or other kind of configurations. These containers are isolated, so it is not necessary any additional code outside the container, so it can be executed anywhere using the same architecture.

The containers are the solution to the problem of moving environments, as can be, moving from the simulator environment to the real use. An application can be tested with certainty that the code is not going to work in a different method, because everything needed by the code is inside the container.

Containers represent a logical packaging mechanism where applications have everything they need to run. Described in a small configuration file called dockerfile. The dockerfile will be enough to adapt the execution environment and configure the server where it will be scaled. From that file, an image can be generated that can be displayed on a server in seconds as shows figure 4.4.2.

The use of containers provides that resources can be isolated, this characteristic is important in this kind of project because every system can be adapted for different versions of the used tools as can be NVIDIA Cuda, OpenCV, ROS, even the operative system, because some tools are running on Ubuntu 18.04 and others on Ubuntu 16.04.

Also Docker provides Docker Hub [2], which is a library and community for container images. Browse over 100,000 container images from software vendors, open-source projects, and the community.

[2] https://hub.docker.com/

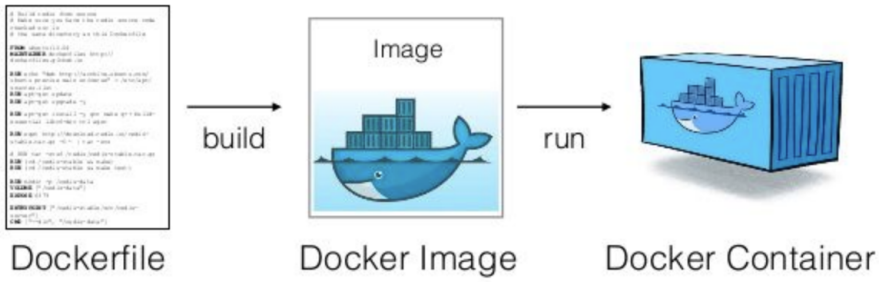


Figure 4.4.2: Docker build from a dockerfiles.

4.5 Git

Git [3] is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

Git was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development. Every Git directory on every computer is a full-fledged repository with complete history and full version-tracking abilities.

The work flow is represented on Figure 4.5.1 where to add a new version to the project the developer has to add the changes, commit them and push to the server. If he only add and commit the changes he will create a new version but on local way.

All the developers involved in this project use Git to have version-control of the different layers of the project, to share the code with the others involved developers Github [4] is used which provides hosting for software development version control using Git.

On Figure 4.5.2 can be seen the process of one developer working on two computers, this is the same structure as two developers working on the same project. OR one developer working from two computers, one in the office and the other at home.

[3] https://git-scm.com/
[4] https://github.com/

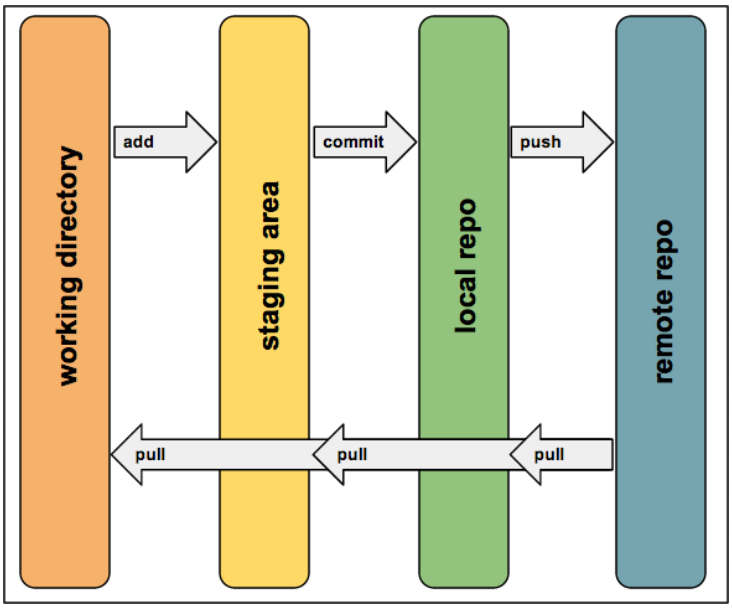


Figure 4.5.1: Git work flow

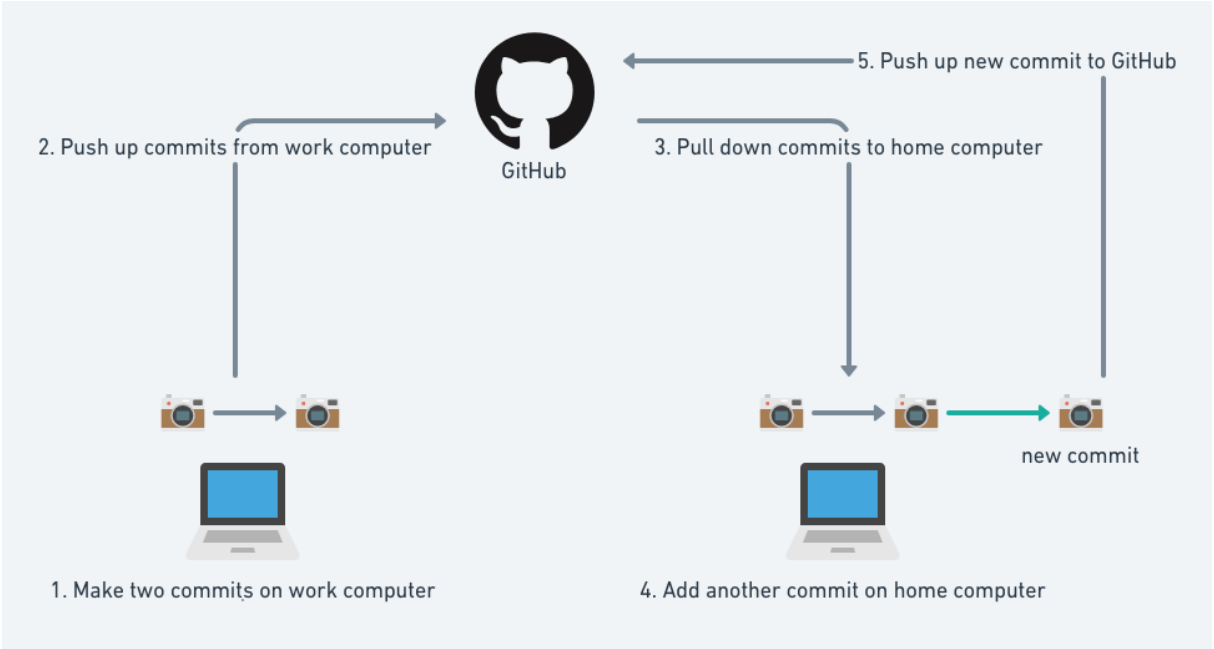


Figure 4.5.2: Version-control process of two computer working on the same project.

4.6 OpenFace 2.0

OpenFace [37] is an open-source tool that implements facial behavior analysis algorithms including: facial landmark detection, head pose tracking, eye gaze and facial Action Unit estimation.

4.6.1 Facial landmark detection and tracking

Openface uses Conditional Local Neural Fields (CLNF) [38] for facial landmark detection and tracking. CLNF is an instance of a Constrained Local Model (CLM) [39], which uses more advanced patch experts and optimization function. It is based in two main components: Point Distribution Model (PDM) that captures landmark shape variations and patch experts that capture local variations of each landmark.

4.6.1.1 Conditional Local Neural Fields (CLNF)

CLNF includes a novel Local Neural Field patch expert that learns the non-linearities and spatial relationships between pixel values and probability of landmark alignment. It also uses a novel Non-uniform Regularised Landmark Mean Shift fitting technique, which take into account patch reliabilities.

4.6.1.1.1 Patch Experts Patch experts evaluate the probability of a landmark being aligned at a particular pixel location.

$$\pi_{x_i} = C_i(x_i; I) \tag{4.1}$$

On Equation 4.1, C_i is the output of the regressor for the i feature at the x_i localization in the image I which gives the response π_{x_i} . The misalignment can then be modeled using a regressor that gives values from 0 (no alignment) to 1 (perfect alignment).

There are different methods proposed as patch experts, Support Vector Regressor (SVR) models [40] and logistic regressors or matching techniques as shows Figure 4.6.1.

The most used expert is the SVR with a logistic regressor. Linear SVR is used because of its computational simplicity

4.6.1.1.2 Local Neural Field patch expert Local Neural Field (LNF) patch expert, brings the non-linearity of Conditional Neural Fields [41] together with the flexibility and continuous output of Continuous Conditional Random Fields [42]. The patch expert can capture relationships between pixels (neighbouring and longer distance) by learning both similarity and long distance sparsity constraints.

4.6.1.2 Landmark detection evaluation

OpenFace has been evaluated and compared with a number of recent baselines in a cross-dataset evaluation setup. The other evaluated systems where used with the code provided by their developers.

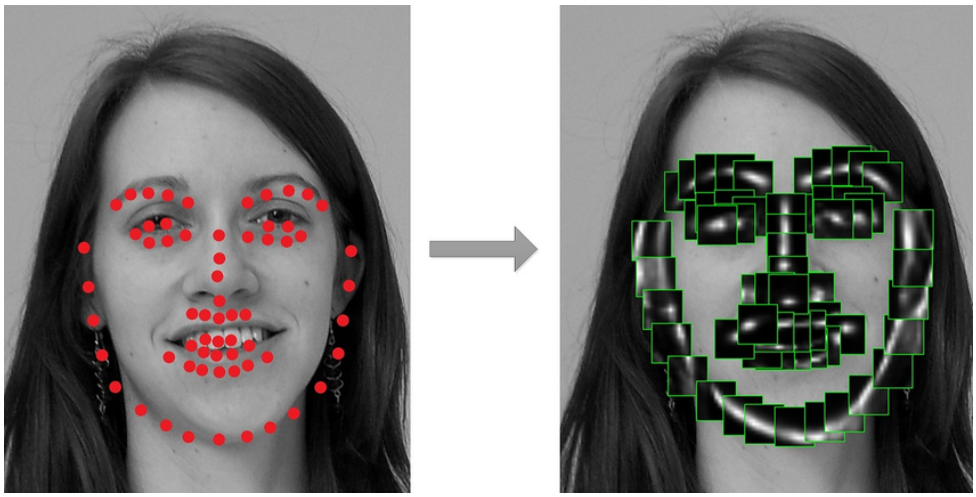


Figure 4.6.1: Example of $11 \times 11px$ SVR patch experts evaluated on a greyscale image of a face at the locations indicated by red circles and a $21 \times 21px$ area of interest surrounding them. The green bounding boxes represent a particular patch expert response map in the area of interest. The darker response values indicate low probability of alignment, and brighter values indicate high probability of alignment.

4.6.1.2.1 Dataset To test the ability to detect landmark by OpenFace, two publicly dataset were used: IJB-FL [43] and 300VW [44]. The first one is a landmark annotated subset of IJB-A [45], a face recognition benchmark that has 180 images (128 frontal and 52 profile faces). The images are non-frontal pose, with heavy occlusion and poor image quality, which make more difficult to pass the benchmark. The second one, contains 64 videos labeled for 68 facial landmarks for every frame. Videos are divided in three subclasses: 1. Laboratory and naturalistic well-lit conditions; 2. Unconstrained conditions, like varied illumination, dark rooms and overexposed frames; 3. Completely unconstrained conditions with occlusions like occlusions with the hand.

4.6.1.2.2 Baselines To test the ability to detect landmark by OpenFace, it has to be compared with other algorithms in the state of the art and that have been trained to detect the same landmarks. Coarsde to Fine Shape Search (CFSS) [46] is a cascaded regression approach. Project-out cascaded regression (PO-CR) [47] is also a cascaded regression approach that updates the shape model parameter rather than predicting landmark localization in a projected-out space.

4.6.1.2.3 Results in IJB-FL can be seen on Figure 4.6.2 and in 300VW on Figure 4.6.3.

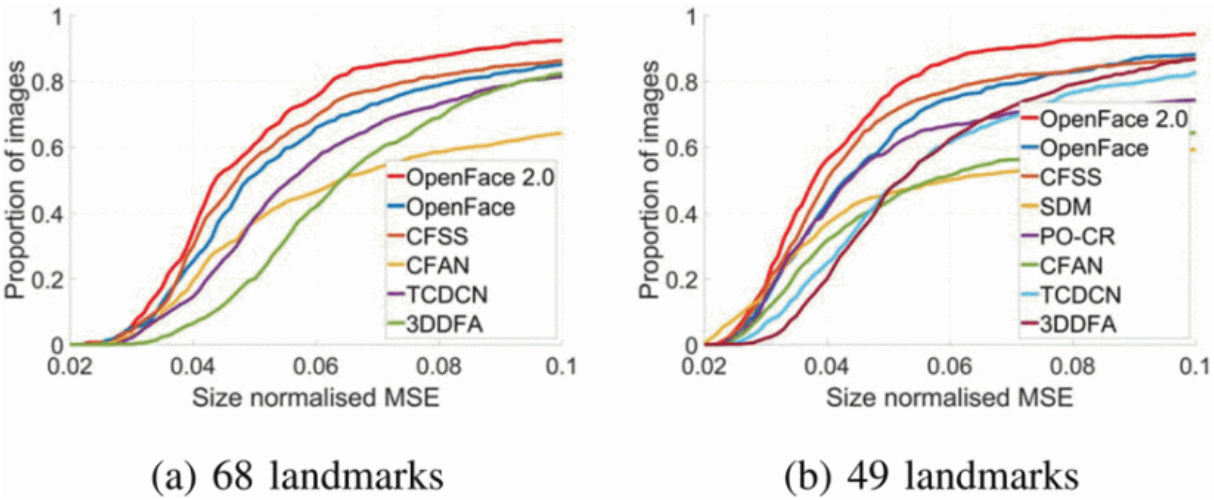


Figure 4.6.2: Fitting on IJB-FL using openface 2.0 and comparing against recent landmark detection methods. None of the approaches were trained on IJB-FL, allowing to evaluate ability to generalize.

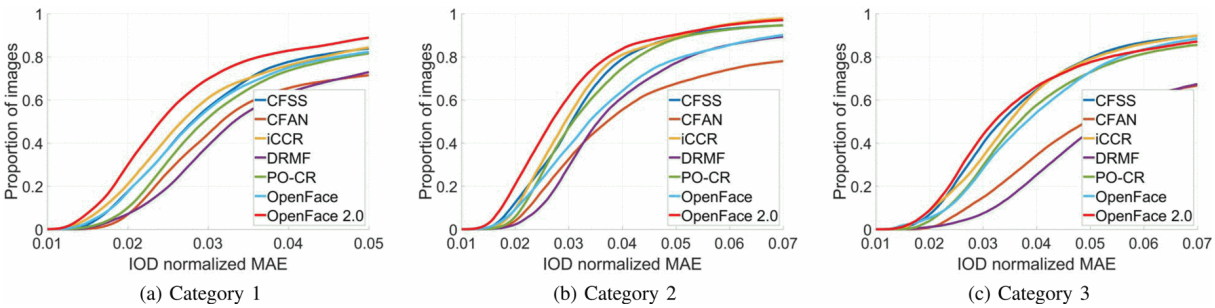


Figure 4.6.3: Fitting on the 300VW dataset using openface 2.0 and recently proposed landmark detection approaches. We only report performance on 49 landmarks as that allows us to compare to more baselines. All of the methods except for iccr were not trained or validated on 300VW dataset.

4.6.2 Head pose estimation

The model is able to extract head pose (translation and rotation of the head). This is possible, because CLNF uses a 3D representation of facial landmark projecting them to the image using orthographic camera projection. With this tool, OpenFace can estimate head pose once the landmark are detected solving the Perspective-n-Point problem [48] defined in Equation 4.2, where $[x\ y\ z\ 1]$ is the homogeneous world point, $[u\ v\ 1]$ is the corresponding homogeneous image point, the first matrix is the matrix of the intrinsic camera parameters (where f_x and f_y are the scaled focal lengths, γ is the skew parameter which normally it is 0 and $u_0\ v_0$ is the principal point) and the second matrix correspond to the rotation and translation matrix (extrinsic parameters).

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{4.2}$$

OpenFace can obtain accurate head pose estimation if the camera calibration parameters are provided (focal length and principal point). In their absence OpenFace uses a rough estimate based on image size to calculate it.

4.6.2.1 Head pose estimation evaluation

OpenFace has been evaluated and compared with a number of recent baselines in a cross-dataset evaluation setup. The other systems evaluated where used with the code provided by their developers.

4.6.2.1.1 Dataset used in this evaluation where provided with ground truth pose data: BU [49] and ICT-3DHP [40]

4.6.2.1.2 Baselines used to comparison are Chedra framework [50], CLM [51], CLM-Z [40], Regression Forests [52] and OpenFace 1.0 [53], which is the first version of the framework used in this work.

4.6.2.1.3 Results on both datasset for all baselines compared with OpenFace 2.0 can be seen on Table 4.1 and 4.2. Results are on pair other method of the state-of-the-art.

Table 4.1: Head pose estimation results on the BU dataset. Measured in mean absolute degree error. Note that BU dataset only contains RGB images so no comparison against CLM-Z and Regression forests was performed.

Method	Yaw	Pitch	Roll	Mean	Median
CLM	3.0	3.5	2.3	2.9	2.0
Chedra	3.8	4.6	2.8	3.8	2.5
OpenFace 1.0	2.8	3.3	**2.3**	2.8	2.0
OpenFace 2.0	**2.4**	**3.2**	2.4	**2.6**	**1.8**

4.6.3 Eye gaze estimation

To estimate eye gaze, OpenFace uses a CLNF landmark detector to detect eyelid, iris and pupil. For training the model, SynthesEyes [54] training dataset was used. Which is a synthesizing perfectly labelled photo-realistic training data in a fraction of the time.

Table 4.2: Head pose estimation results on ICT-3DHP. Measured in mean absolute degree error.

Method	Yaw	Pitch	Roll	Mean
Reg. Forests	7.2	9.4	7.5	8.0
CLM-Z	5.1	3.9	4.6	4.6
CLM	4.8	4.2	4.5	4.5
Chedra	13.9	14.7	10.3	13.0
OpenFace 1.0	3.6	3.6	3.6	3.6
OpenFace 2.0	**3.1**	**3.5**	**3.1**	**3.2**

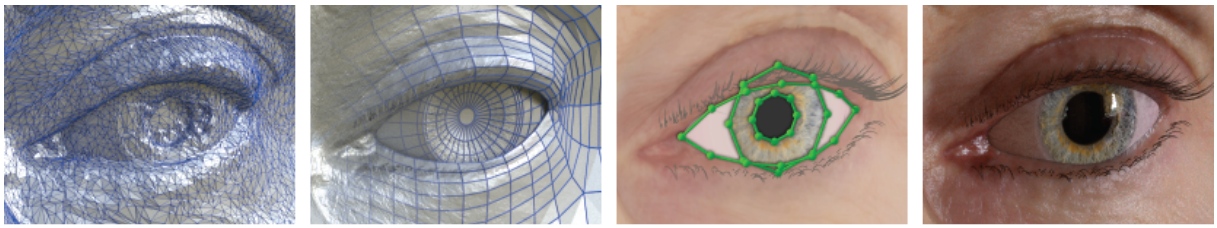


Figure 4.6.4: Photorealistic images of eyes for use as training data.

With the pupil detected and eye, OpenFace computes the eye gaze vector for each eye. OpenFace fires a ray from the camera to the center of the pupil in the image plane and computes it's intersection with the eye-ball sphere. This intersection gives the pupil localization in 3D camera coordinates and the vector from the 3D eye-ball center to the pupil localization, which is the gaze vector for each eye.

According to the authors of OpenFace, this is a fast and accurate method for person independent eye-gaze estimation.

4.6.3.1 Eye gaze estimation evaluation

OpenFace has been evaluated and compared with a number of recent baselines in a cross-dataset evaluation setup. The systems other evaluated where used with the code provided by their developers.

4.6.3.1.1 Dataset OpenFace has been evaluated on a challenging dataset called MPI-IGaze [55] to evaluate eye gaze estimation. This dataset was recorded in realistic laptop scenarios. OpenFace has been evaluated in 750 images from the dataset.

4.6.3.1.2 Baselines used on this comparation has not to be trained on this dataset in order to get validated results. They compared their model to a CNN proposed on [56], Eyelab geometry based model [57] and a kNN approach based on the UnityEyes datset [58].

4.6.3.1.3 Results can be seen as an absolute mean degree error on Table 4.3. Results are on the state-of-the-art performance.

Table 4.3: Results comparing our method to previous work for cross dataset gaze estimation on MPIIGaze , measure in mean absolute degree error.

Model	Gaze error (^{o})
EyeTab	47.1
CNN on UT	13.91
CNN on SYntheEyes	13.55
CNN on SYntheEyes + UT	11.12
OpenFace 1.0	9.96
UnityEyes	9.95
OpenFace 2.0	**9.10**

4.6.4 Facial expression recognition

OpenFace can also recognizes facial expression throught detecting facial Action Unit (AU) intensity and presence. It uses a method based on AU recognition framework [59]. This framework has been adapted to work better with video sequences.

Table 4.4: List of AUs in OpenFace.

AU	Name
AU01	INNER BROW RAISER
AU02	OUTER BRW RAISER
AU04	BROW LOWERER
AU05	UPPER LID RAISER
AU06	CHEEK RAISER
AU07	LID TIGHTENER
AU09	NOSE WRINKLER
AU10	UPPER LIP RAISER
AU12	LIP CORNER PULLER
AU14	DIMPLER
AU15	LIP CORNER DEPRESSOR
AU17	CHIN RAISER
AU20	LIP STRETCHED
AU23	LIP TIGHTENER
AU25	LIPS PART
AU26	JAW DROP
AU28	LIP SUCK
AU45	BLINK

As features it uses the concatenation of dimesionality reduced HOGs (Histogram of Oriented Gradients) [60] from similarity aligned 112 x 112 pixel face image and facial shape

features. In order to get the data for a video sequenceS, the median value is calulated for the frame.

The model has been trained on DISFa [61], SEMAINE [62], BP4D [63], UNBC-McMaster [64], Bosphorus [65] and FERA 2011 [66] datasets.

Figure 4.6.5 shows how are displayed the Action Units while the toll is running.

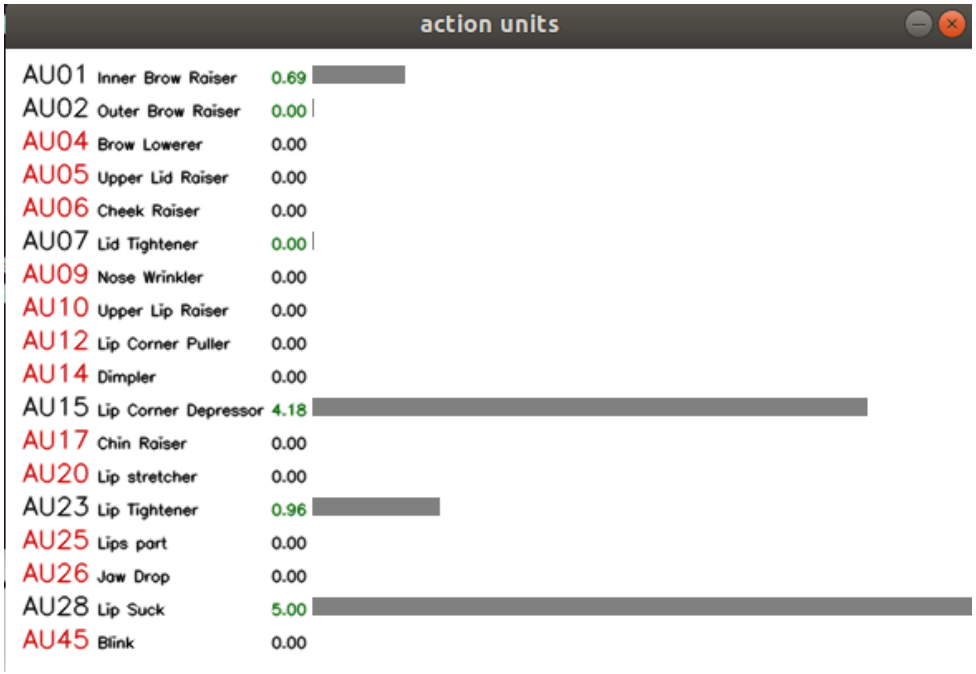


Figure 4.6.5: List of AUs detected by OpenFace.

4.6.4.1 Action Unit recognition evaluation

OpenFace has been evaluated and compared with a number of recent baselines in a cross-dataset evaluation setup.

4.6.4.1.1 Dataset evaluated to offer a comparison between different baselines and called DISFa [61]

4.6.4.1.2 Baselines used on this section are general methods where there are not new free tools, only a few commercial that does not allow a public comparison with other tools.

Continuous Conditional Neural Fields (CCNF)[67] model is a temporal approach for AU intensity estimation based on non-negative matrix factorization features around facial landmark points. Iterative Regularized Kernel Regression IRKR [68] is a recently proposed kernel learning method for AU intensity estimation. It is an iterative nonlinear

feature selection method with a Lasso-regularized version of Metric Regularized Kernel Regression. A generative latent tree (LT) [69].

Also they included on the validation two CNN baselines [70] and [71]

4.6.4.1.3 Results can be seen on Table 4.5. Comparing our model to baselines on the DISFA dataset, results are reported as Pearson Correlation Coefficient. It can be seen that an SVR-HOG approach employed by OpenFace 2.0 outperforms the more complex and recent approaches for AU detection on this challenging dataset.

Table 4.5: Action Unit recognition evaluation. Comparing our model to baselines on the DISFA dataset, results reported as Pearson Correlation Coefficient.

Method	IRKR	LT	CNN	D-CNN	CCNF	OpenFace 2.0
AU1	0.1	0.41	0.60	0.49	0.48	**0.64**
AU2	**0.68**	0.44	0.53	0.39	0.50	0.50
AU4	0.68	0.50	0.64	0.62	0.52	**0.70**
AU5	0.49	0.29	0.38	0.44	0.48	**0.67**
AU6	**0.65**	0.55	0.55	0.53	0.45	0.59
AU9	0.43	0.32	0.59	0.55	0.36	**0.54**
AU12	0.83	0.76	**0.85**	**0.85**	0.70	**0.85**
AU15	0.34	0.11	0.22	0.25	**0.41**	0.39
AU17	0.35	0.31	0.37	0.41	0.39	**0.49**
AU20	0.21	0.16	0.15	0.19	0.11	**0.22**
AU25	0.86	0.82	0.88	0.87	**0.89**	0.85
AU26	0.62	0.49	0.60	0.59	0.57	**0.67**
Mean	0.57	0.43	0.53	0.51	0.49	**0.59**

4.7 Kalman Filter

Kalman Filter [72] presented by RE Kalman in 1960 is an algorithm that predicts the output of a signal based on a series of measurements. This filter allows to have a more accurate measure based on a series of measurements and taking into account the sequence of the signal instead only one measure.

To achieve this purpose some steps have to be done. Firstly a predict step:

$$\begin{aligned} \hat{x}_{k|k-1} &= F_k \hat{x}_{k-1|k-1} + B_k u_k \\ P_{k|k-1} &= F_k P_{k-1|k-1} F_k^T + Q_k \end{aligned} \tag{4.3}$$

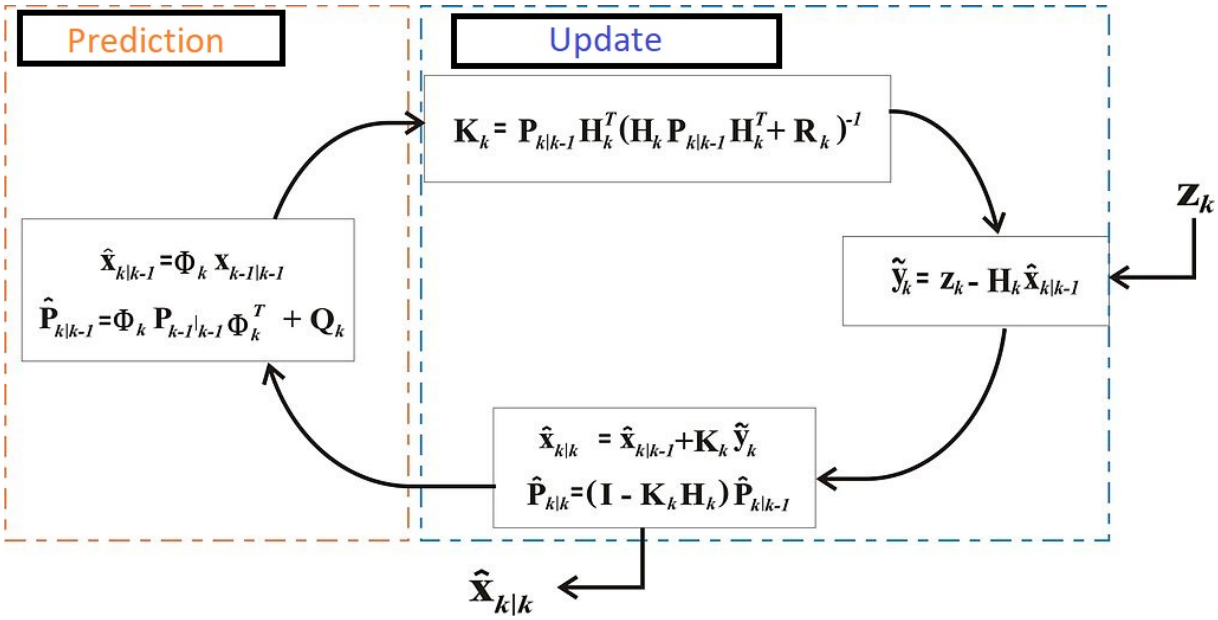


Figure 4.7.1: Kalman filter

Where F_k is the state transition model which is applied to the previous state $\hat{x}_{k-1|k-1}$; B_k is the control-input model which is applied to the control vector u_k. $P_{k|k-1}$ is predicted (a priori) estimate covariance and Q_k is the covariance of the process.

And a second step called Update:

$$\begin{aligned} \tilde{y}_k &= z_k - H_k\hat{x}_{k|k-1} \\ S_k &= H_k P_{k|k-1} H_k^T + R_k \\ K_k &= P_{k|k-1} H_k^T S_k^{-1} \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k\tilde{y}_k \\ P_{k|k} &= (I - K_k H_k) P_{k\bar{c}1} \\ \tilde{y}_{k|k} &= z_k - H_k\hat{x}_{k|k} \end{aligned} \tag{4.4}$$

H_k is the observation model which maps the true state space into the observed space; R_k is the covariance of the measure. $\tilde{y}_k$ is the innovation or measurement pre-fit residual; S_k is the innovation (or pre-fit residual) covariance; K_k is the optimal Kalman gain $\hat{x}_{k|k}$ is the updated (a posteriori) state estimate; $P_{k|k}$ is the Updated (a posteriori) estimate covariance and $\tilde{y}_{k|k}$ is the measurement post-fit residual.

4.8 QT Creator

Qt Creator is a cross-platform, integrated development environment (IDE) for application developers to create applications for multiple desktop, embedded, and mobile device

platforms.

Figure 4.8.1: Qt Creator.

Qt Creator allows the developer to design, develop and deploy GUI (Graphical User Interface). GUIs are very important for HMI (Human Machines Interfaces) because they allow the user to watch all the information relevant at a glance, which is important when you are driving because you have to keep your eyes on the road.

4.9 CARLA simulator

CARLA [73] has been developed from the ground up to support development, training, and validation of autonomous driving systems. In addition to open-source code and protocols, CARLA provides open digital assets (urban layouts, buildings, vehicles) that were created for this purpose and can be freely used. The simulation platform supports flexible specification of sensor suites, environmental conditions, full control of all static and dynamic actors, maps generation and much more.

Figure 4.9.1: Carla simulator

CARLA also provides a ROS-bridge, which enables two-way communication between ROS and CARLA. The information from the CARLA server is translated to ROS topics.

In the same way, the messages sent between nodes in ROS are translated to commands to be applied in CARLA.

Chapter 5

Development

5.1 Introduction

This chapter describes the systems carried out in this work. This will is formed of two section, one for the gaze tracking and another for the HMI (Human Machine Interface) implemented in Techs4AgedCar.

5.2 Gaze focalization

This section introduces the development of the tools used to test OpenFace as a cheap and non-intrusive gaze tracker used to estimate the visual driver attention. Two different approximation where done to transform the output of OpenFace to the gaze focalization on the driving scene.

This work presents a low-cost and non-intrusive camera-based gaze mapping system able to estimate the visual driver attention on challenging pre-recorded traffic scenes through a heat map. Gaze direction is calculated by using the open-source state-of-the-art OpenFace 2.0 Toolkit [1]. After a slight calibration process and using a simple projection model, a heat attention map is obtained. Our goal is to implement a cheap and user-friendly measurement system able to work as most sophisticated systems in the focalization of accidents over complex scenarios. Our proposal has been validated by using the recent and challenging public dataset DADA2000 which collects annotated driving scenarios based on real accidents providing the accident position and the attention map generated by some users who have watched the scene. We present some performance results for our attention model and we compare our numbers with the obtained with an expensive and active desktop-mounted eye-tracker [2] in similar conditions that the reported by the authors, reaching on par results. This fact confirms our technique is a good tool for driver attention monitoring able to be used in the design of take over and

driving environment awareness systems for automated vehicles. From our knowledge, this is the first time a camera-based vehicle-mounted driver attention system is evaluated in the challenging DADA2000 dataset.

The complete environment is composed by three main subsystems. The first one is the one that launches the camera in this case we have used the ZED stereo camera because facilities to work with ROS, but as a monocular camera. Another reason to choose this camera over a standard web cam was its configurations, that allows the developer to change multiple parameters in order to get the better performance. The second one is the Openface framework using ROS that allows the tool to publish on a topic all the information extracted by it. And finally, the third subsystem is where the information of OpenFace is extracted and used to achieve the task. This subsystem is considered as a calibration method in order to achieve better results.

These three subsystems are the main part of the proposed system but also there are some test that haven been done after these subsystems, like the precision task or the use in the DADA dataset.

In the next subsections the two calibration methods developed in this work are introduced.

5.2.1 Linear calibration

Firstly, a linear approximation was done to test the tool as a gaze tracker in road environments. OpenFace provides some parameters about the face tracked that we use. Figure 5.2.1 shows how the developed framework is in this calibration method and how are the three main subsections connected to work together as a principal system.

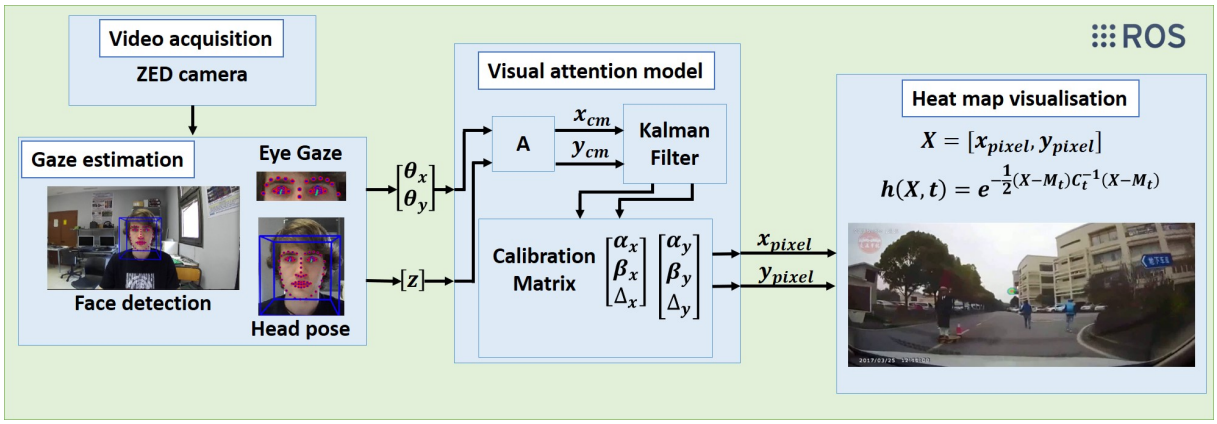


Figure 5.2.1: Linear calibration framework.

In our application gaze vector is used to know where the person under the test is looking. Although the human under test has two eyes and generates two vectors one for each eye, OpenFace provides a single vector which is the fusion of the two eyes. This

vector is given as two angles, $[\theta_x, \theta_y]$, regarding the (X,Y) axis in the camera reference system as shows Figure 5.2.2. This vector is projected on the screen in order to get an attention pixel through a visual model. To minimize gaze vector uncertainty a Kalman Filter, experimentally adjusted, is implemented over the projection.

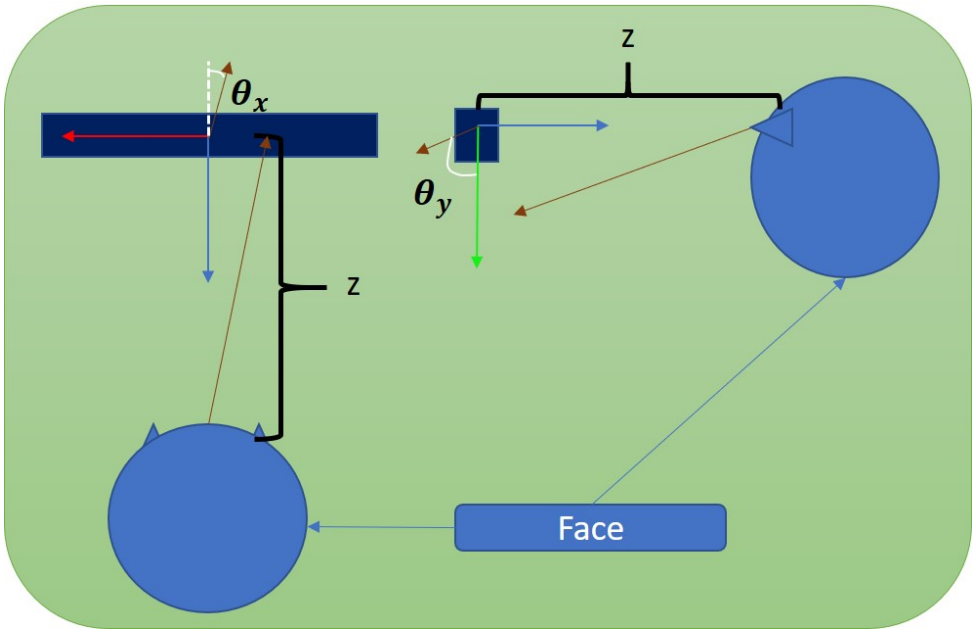


Figure 5.2.2: OpenFace gaze angles explanation.

The user is in front of a 24" screen at 60 centimetres from it. Then, his visual field varies between ±27 degrees on the X axis and 30 degrees on the Y axis. In these ranges the projection model is quite linear. This is the reason because we propose to use a simple projection model learned in a previous calibration step in the same way that the done in a touch screen calibration process [74].

We apply a slight calibration method at the beginning of each experiment that consist on looking at four points on the screen to get the projection matrix and the limits of the screen where the test is run. Points are displayed on the screen during 5 seconds but to obtain an accurate calibration the first second is discarded because it is the time that the user needs to change the look from one point to other in order to get also the Kalman accommodated.

Calibration method translates the gaze vector into coordinates that accurately represent the projection of this vector on the screen in pixels. For each calibration point, the gaze vector ($[\theta_x, \theta_y]$) is projected on the screen ($[X'_k, Y'_k]$) using the trigonometric equations shown in Equation 5.2. z is the average distance of the user's eyes regarding the camera reference. This parameter is also provided by OpenFace.

$$z = \frac{z_{right\_eye} + z_{left\_eye}}{2} \tag{5.1}$$

$$\begin{pmatrix} X'_k \\ Y'_k \end{pmatrix} = \begin{pmatrix} -z\ sin(\theta_x) \\ z\ sin(\theta_y) \end{pmatrix} \tag{5.2}$$

The matrix A is formed by the projected points (in cm) for the 4 points displayed on camera reference. In this step, these points have to be matched with the known positions of the calibration points on the screen in pixels ($[X_k, Y_k]$) regarding the screen reference (top-left corner), which are: $[X_1, Y_1] = [200, 200]$, $[X_2, Y_2] = [200, 900]$, $[X_3, Y_3] = [1700, 900]$, $[X_1, Y_1] = [1700, 200]$.

$$A = \begin{pmatrix} X'_1 & Y'_1 & 1 \\ X'_2 & Y'_2 & 1 \\ X'_3 & Y'_3 & 1 \\ X'_4 & Y'_4 & 1 \end{pmatrix} \tag{5.3}$$

To match these data we use a parametric model composed of the coefficients $[\alpha_X, \beta_x, \Delta_x]$ and $[\alpha_y, \beta_y, \Delta_y]$, which transform the spatial space from centimetres to pixel and change the reference from the camera to the left upper corner of the screen, according to the equation 5.4 .

$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix} = A \begin{pmatrix} \alpha_X \\ \beta_X \\ \Delta_X \end{pmatrix} and \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{pmatrix} = A \begin{pmatrix} \alpha_Y \\ \beta_Y \\ \Delta_Y \end{pmatrix} \tag{5.4}$$

Equation 5.5 shows how the coefficients are calculated from equation 5.4.

$$\begin{pmatrix} \alpha_X \\ \beta_X \\ \Delta X \end{pmatrix} = A^{-1} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix} and \begin{pmatrix} \alpha_Y \\ \beta_Y \\ \Delta Y \end{pmatrix} = A^{-1} \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{pmatrix} \tag{5.5}$$

After calibration, each attention point is obtained applying the trigonometric projection of its corresponding vector gaze provided by OpenFace, where $X_{cm} = -z \cdot sin(\theta_x)$ and $Y_{cm} = z \cdot sin(\theta_y)$, and the parametric transformation carried out by Equation 5.6.

$$\begin{aligned} x_{pixel} &= \alpha_X X_{cm} + \beta_X Y_{cm} + \Delta_X \\ y_{pixel} &= \alpha_Y X_{cm} + \beta_Y Y_{cm} + \Delta_Y \end{aligned} \tag{5.6}$$

To perform this calibration the equation has to be solved, in order to achieve that

the solution can be derived from Cramer's rule. Where n is the number of calibration points, X_k and Y_k are the points in pixels and $X_k^{'}$ and $Y_k^{'}$ are the points measured with the triangulation approximation.

$$\begin{aligned} \alpha_X = \frac{\Delta_{x1}}{\Delta}; \beta_X = \frac{\Delta_{x2}}{\Delta}; \Delta_X = \frac{\Delta_{x3}}{\Delta} \\ \alpha_Y = \frac{\Delta_{y1}}{\Delta}; \beta_Y = \frac{\Delta_{y2}}{\Delta}; \Delta_Y = \frac{\Delta_{y3}}{\Delta} \end{aligned} \tag{5.7}$$

where

$$\begin{aligned} \Delta = n \cdot (a \cdot b - c^2) + 2 \cdot c \cdot d \cdot e - a \cdot e^2 - b \cdot d^2 \\ \Delta_{x1} = n \cdot (\mathbf{X}_1 \cdot b - \mathbf{X}_2 \cdot c) + e \cdot (\mathbf{X}_2 \cdot d - \mathbf{X}_1 \cdot e) + \mathbf{X}_3 \cdot (c \cdot e - b \cdot d) \\ \Delta_{x2} = n \cdot (\mathbf{X}_2 \cdot a - \mathbf{X}_1 \cdot c) + d \cdot (\mathbf{X}_1 \cdot e - \mathbf{X}_2 \cdot d) + \mathbf{X}_3 \cdot (c \cdot d - a \cdot e) \\ \Delta_{x3} = \mathbf{X}_3 \cdot (a \cdot b - c^2) + \mathbf{X}_1 \cdot (c \cdot e - b \cdot d) + \mathbf{X}_2 \cdot (c \cdot d - \cdot e) \\ \Delta_{y1} = n \cdot (\mathbf{Y}_1 \cdot b - \mathbf{Y}_2 \cdot c) + e \cdot (\mathbf{Y}_2 \cdot d - \mathbf{Y}_1 \cdot e) + \mathbf{Y}_3 \cdot (c \cdot e - b \cdot d) \\ \Delta_{y2} = n \cdot (\mathbf{Y}_2 \cdot a - \mathbf{Y}_1 \cdot c) + d \cdot (\mathbf{Y}_1 \cdot e - \mathbf{Y}_2 \cdot d) + \mathbf{Y}_3 \cdot (c \cdot d - a \cdot e) \\ \Delta_{y3} = \mathbf{Y}_3 \cdot (a \cdot b - c^2) + \mathbf{Y}_1 \cdot (c \cdot e - b \cdot d) + \mathbf{Y}_2 \cdot (c \cdot d - \cdot e) \\ a = \sum_{k=1}^{n} X_k^{'2};\ b = \sum_{k=1}^{n} Y_k^{'2};\ c = \sum_{k=1}^{n} (X_k^{'} \cdot Y_k^{'});\ d = \sum_{k=1}^{n} X_k^{'};\ e = \sum_{k=1}^{n} Y_k^{'}; \\ \mathbf{X}_1 = \sum_{k=1}^{n} (X_k^{'} \cdot X_k);\ \mathbf{X}_2 = \sum_{k=1}^{n} (Y_k^{'} \cdot X_k);\ \mathbf{X}_3 = \sum_{k=1}^{n} X_k \\ \mathbf{Y}_1 = \sum_{k=1}^{n} (X_k^{'} \cdot Y_k);\ \mathbf{Y}_2 = \sum_{k=1}^{n} (Y_k^{'} \cdot Y_k);\ \mathbf{Y}_3 = \sum_{k=1}^{n} Y_k \end{aligned} \tag{5.8}$$

The calibration method proposed has been developed in a C++ script using ROS to communicate with the other modules. Some test where done in order to validate this system that will be explained later.

5.2.2 Non-Linear calibration

Angles gaze are not linear and they only can be assumed as linear when the gaze range is small. However, if we are looking for a general case the system has to evolve to a next level. With this premise, we propose to implement a Nonlinear AutoRegressive Moving Average model with eXogenous inputs (NARMAX model). Because the output has to have two parameters one for the X coordinate and another for the Y coordinate, two models will be calculated each time the calibration method is done.

This system substitutes completely the linear system proposed in previous section, because it is going to change all the methods implemented by the linear approximation.

It is composed by three main parts as shows Figure 5.2.3, which represents the framework used on this calibration method, and how the subsystems are communicated among them in order to achieve the best result. The first one is in charge of getting data to calibrate. Second one calculates the regressors for the model. And finally, the last one uses these regressors and sends the topic to the ROS environment.

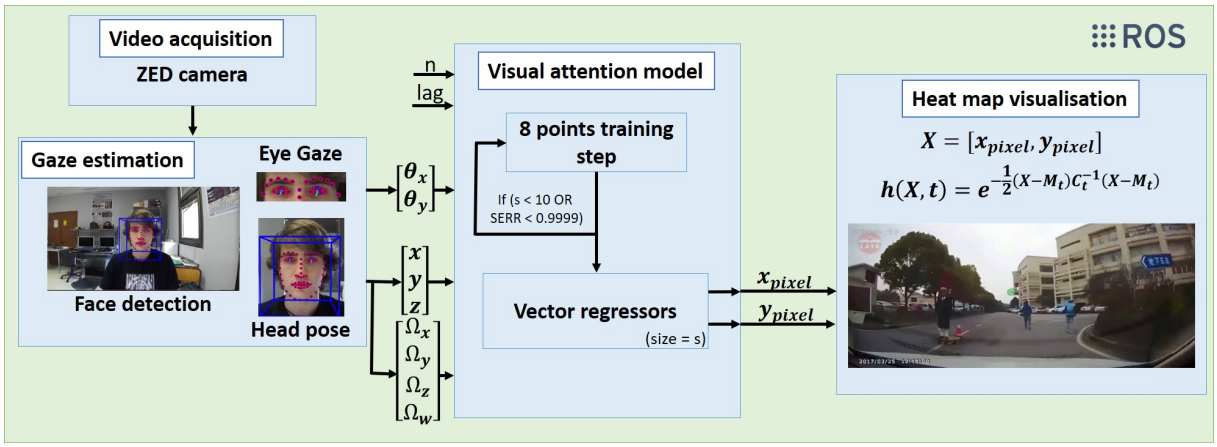


Figure 5.2.3: Non-linear calibration framework.

In order to achieve this calibration method it is necessary a training method. So, at the beginning of any test the user has to look eight points around the screen. Points are displayed during 8 seconds but in order to get a better accuracy on the model the first two seconds are not recorded because in this time eyes are in transition between fixations.

To perform this task, the eight points are displayed over a white image using OpenCV. These points have a 30 pixel radio. These points are placed on $[X_1, Y_1] = [200, 200]$, $[X_2, Y_2] = [700, 200]$, $[X_3, Y_3] = [1100, 200]$, $[X_4, Y_4] = [1700, 200]$, $[X_5, Y_5] = [200, 700]$, $[X_6, Y_6] = [700, 700]$, $[X_7, Y_7] = [1100, 700]$ and $[X_7, Y_8] = [1700, 700]$. These eight points will be evaluated in a posterior part to test the model.

Data is composed by the data provided by OpenFace and the point where the user is looking at. Data provided by OpenFace on the training are:

- Gaze angle X (θ_x)
- Gaze angle Y (θ_y)
- Head position X (x)
- Head position Y (y)
- Head position Z (z)
- Head rotation X (Ω_x)
- Head rotation Y (Ω_y)

- Head rotation Z (Ω_z)
- Head rotation W (Ω_w)

Data is referenced to the camera. Data is recorded on a *.txt* file with the next structure: Gaze angle X; Gaze angle Y; Head position X; Head position Y; Head position Z; Head rotation X; Head rotation Y; Head rotation Z; Head rotation W; X pixel; Y pixel.

This file created with the data recorded will be charged on the next step of the model where it is going to be trained.

5.2.2.1 NARMAX model

With the obtained data next step is to get the model. This part has been designed as an external tool but will be integrated on the ROS environment on future work in order to automatize all the process. The training method has been developed on Python because of the facility of this language to work with math models (Figure 5.2.4).

NARMAX model is fit with these parameters, this does not mean that has to use all of them, the model will use the ones that fit better. After some experiments with the model the best way to get an accurate model without spending so much time is to train until Error Reduction Ratio (ERR) is more than 0.9999 or 10 regressors has been calculated. More regressors complicate the model with a slight improvement on accuracy and over-fitting the model.

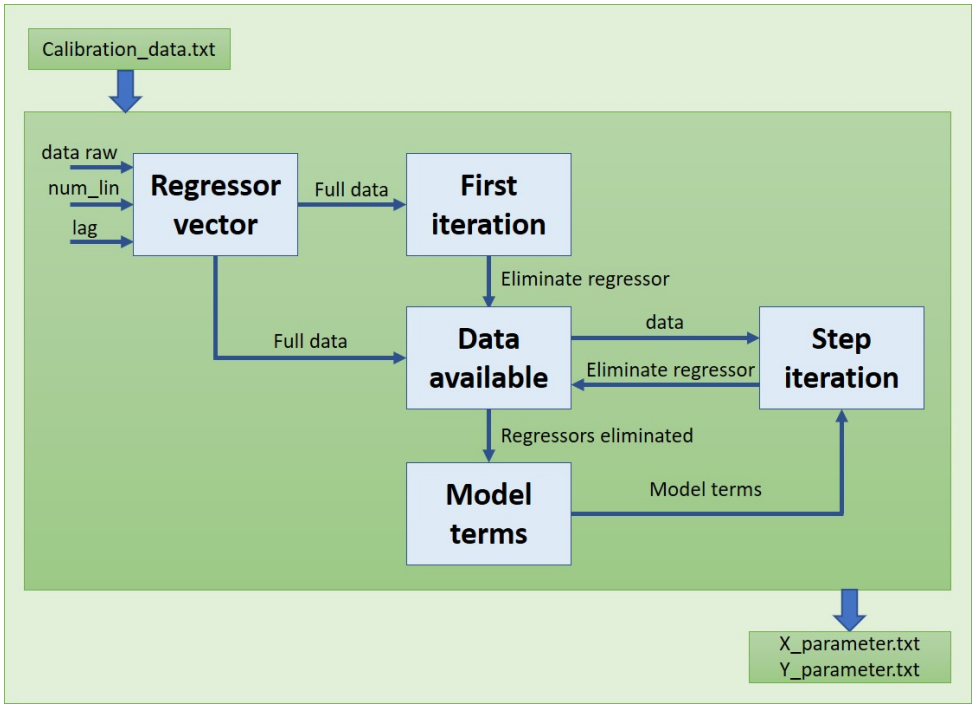


Figure 5.2.4: Non-linear calibration python script flow.

At the beginning some problems were found, firstly the model consider previous outputs as parameters for the model. We had to change the way we calculated the model because with this calibration previous outputs were the same when you were looking at the same point, so the model was only considering this parameter.

5.2.3 Heat map visualization

In order to visualize the results a heat map is built in real time while the test is running. To make this possible a temporal window of 1 second is recorded to calculate the mean and the standard deviation needed to obtain the probability for each Gaussian.

With these two probabilities a global probability is calculated for each pixel in order to get the heat map. To be more efficiency, because the exponential has a a great impact in the computer efficiency, some phases were tested to get the best results.

Firstly, a function based on threshold was made in order to only calculate some logarithms at the beginning of the test, which allow the code to be faster. But the authors were not satisfied with the results because the map was too poor on color scale because only some colors were represented.

To make computationally possible the task other solution was implemented. Instead of calculate the probability of each pixel for the complete image which means 2.073.600 pixels in a 1920x1080 image with two probabilities for each pixel, the calculus is only done on an ellipse with a radius of three times the standard deviation for each axis and centred on the mean.

With this implementation a real time task and with a complete colour scale was built.

To build the heat image OpenCv library[1] was used. It is necessary to build a RGB image from a probability, to make this possible a function is done which calculates the three different values of the the three color channel with the probability obtained in the previous phase.

If the probability goes between 0 and 0.5 the color is built as follows:

$$\begin{aligned} blue &= 255 - 255(2p) \\ green &= 255(2p) \\ red &= 0 \end{aligned} \tag{5.9}$$

[1] https://opencv.org/

If the probability goes between 0.5 and 1 the image is built:

$$\begin{aligned} blue &= 0 \\ green &= 255(2p) \\ red &= 255(2p) - 255 \end{aligned} \tag{5.10}$$

Figure 5.2.5 shows how the heat map is built using the formulas above. Showing the variety of the three colour channels as long as the probability of the heat map increase.

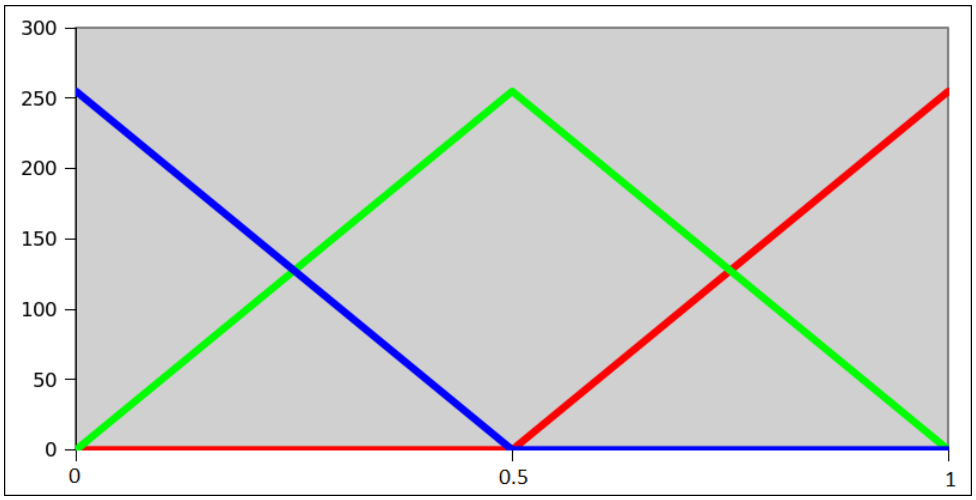


Figure 5.2.5: RGB construction from the probability

To represent the spatial distribution of the gaze fixations on the screen, we use the common visualization approach known as heat map. To take into account the trajectory of the gaze, we integrate the spatial distribution of the fixations within a temporal window [t-w,t], where t is the current frame and w the number of last frames. This temporal window has been set to 1 second as done by [20]. In this way, we have 60 samples for frame in order to generate the heat map. With these samples a Gaussian model is built to represent the different areas of colour of the heat map, where ($M_t = [m_x(t), m_y(t)]$) is the centre of the Gaussian, C_t is the covariance matrix and $h(X,t)$ is the intensity of the heat map for the pixel (X=[x,y]). In a practical way, we draw ellipses in the heat map until three times the standard deviation of the Gaussian model.

Next equation shows how the method is done in a vector way:

$$h(X,t) = e^{-\frac{1}{2}(X-M_t)^T C_t^{-1}(X-M_t)} \tag{5.11}$$

5.3 Vehicle to user layer

In this section we introduce the development of the HMI (Human Machine Interface) for the autonomous car developed in the University of Alcalá called Techs4AgeCar (Figure

5.3.1). This system allows the driver to have access to data car and control the autonomous navigation of the vehicle.

This section is divided in two subsection which will explain how the two different screens placed on the car work and how were developed the tool that control them.

Figure 5.3.1: Actual driver look of the tow screen that composed the vehicle to user layer of the Tech4AgeCar.

5.3.1 Information HMI

This subsystem of the HMI is in charge of showing information to the driver to view data from the car on an intuitive way that enables a safe drive when the car is in manual mode.

This UI (User Interface) was completely developed on Qt Creator using C++ and Qt for the interface layer. Also, to communicate with the other systems on the car has to be compatible with ROS ecosystem. The app is made in different submodules that are communicated. One is in charge of the ROS ecosystem and is the one that gets all the data from ROS topics. This data is send to the UI as signals that have to be lead to the different slots of the implemented widgets. And finally the driver can see and understand the information displayed on the screen with a simple glance.

This system is in constant development because when other layers improve their capabilities, more information can be displayed, also when other developers are worked on

the car allows them to debug the system because some variables can be displayed on the screen. Figure 5.3.2 shows how is the framework built in this section.

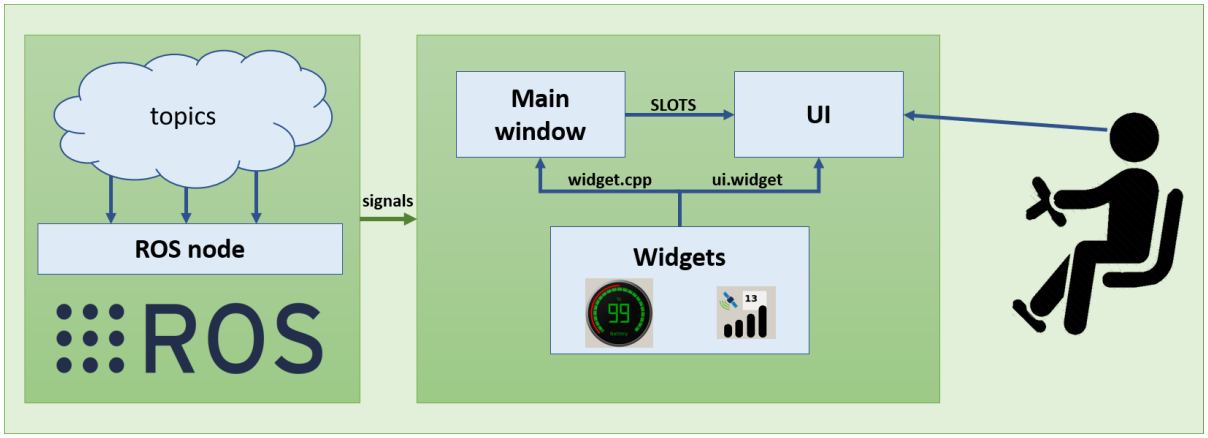


Figure 5.3.2: Framework HMI.

5.3.1.1 Speed gauge

Knowing the vehicle speed is really important at driving because it allows the driver to know the actual speed of the car and enables him to actuate on the car by the throttle or the brake to drive on an adequate speed.

Current velocity is obtained by a topic called */localization/odometry_pose*, which is calculated by the odometry installed in the rear wheels of the car. This message is a *nav_msgs::Odometry* message, which structure can be seen at Figure 5.3.3.

Speed has to be transformed from *m/s* to *km/h* in order to display it like other vehicles do. So it is multiplied by 3.6 and rounded to the nearest integer. This value is sent as a *Q_signal* which will be connected to the ui.speed_gauge by the slot *setValue* that is an integer.

Some parameters of this Gauge where adjusted to match with the characteristics of the vehicle. The max speed was set to 120 km/h. The start angle was set to -45 and the end angle to 225 because we want a clockwise spin.

5.3.1.2 Battery gauge

Knowing the battery range is a required characteristic that must be on the HMI, every driver should know the range of his vehicle because it will allow him to stop and charge his car when the battery range is low. Also this is very important on an electric vehicle because the batteries must not be completely discharged because they will deteriorate further.

```
Header header
   uint32 seq
   time stamp
   string frame_id
string child_frame_id
geometry_msgs/PoseWithCovariance pose
   geometry_msgs/Pose pose
      geometry_msgs/Point position
         float64 x
         float64 y
         float64 z
      geometry_msgs/Quaternion orientation
         float64 x
         float64 y
         float64 z
         float64 w
   float64[36] covariance
geometry_msgs/TwistWithCovariance twist
   geometry_msgs/Twist twist
      geometry_msgs/Vector3 linear
         float64 x
         float64 y
         float64 z
      geometry_msgs/Vector3 angular
         float64 x
         float64 y
         float64 z
   float64[36] covariance
```

Figure 5.3.3: *nav_msgs::Odometry* message structure

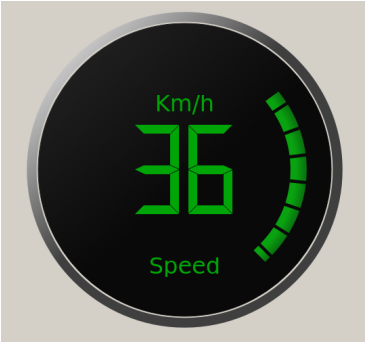


Figure 5.3.4: Speed gauge

Battery is calculate by the BMS (Battery Management System) and published in ROS as a *sec_msgs::BatteryState* which structure can be seen in Figure 5.3.5. The topic name is */car_state/battery*. The value that has to send to the UI is the *percentage*.

Some parameters of this Gauge where adjusted to match with the characteristics of the vehicle. The max value is 100 %. The start angle was set to 225 and the end angle to -45 because we want a counter clockwise spin, to be the opposite of the speed gauge in order to get a symmetry between both gauges. When the battery is below 50 percentage the gauge will turn on red to warm the driver that should charge the battery.

```
uint8 POWER_SUPPLY_STATUS_UNKNOWN=0
uint8 POWER_SUPPLY_STATUS_CHARGING=1
uint8 POWER_SUPPLY_STATUS_DISCHARGING=2
uint8 POWER_SUPPLY_STATUS_NOT_CHARGING=3
uint8 POWER_SUPPLY_STATUS_FULL=4
uint8 POWER_SUPPLY_HEALTH_UNKNOWN=0
uint8 POWER_SUPPLY_HEALTH_GOOD=1
uint8 POWER_SUPPLY_HEALTH_OVERHEAT=2
uint8 POWER_SUPPLY_HEALTH_DEAD=3
uint8 POWER_SUPPLY_HEALTH_OVERVOLTAGE=4
uint8 POWER_SUPPLY_HEALTH_UNSPEC_FAILURE=5
uint8 POWER_SUPPLY_HEALTH_COLD=6
uint8
POWER_SUPPLY_HEALTH_WATCHDOG_TIMER_EXPIRE=7
uint8
POWER_SUPPLY_HEALTH_SAFETY_TIMER_EXPIRE=8
uint8 POWER_SUPPLY_TECHNOLOGY_UNKNOWN=0
uint8 POWER_SUPPLY_TECHNOLOGY_NIMH=1
uint8 POWER_SUPPLY_TECHNOLOGY_LION=2
uint8 POWER_SUPPLY_TECHNOLOGY_LIPO=3
uint8 POWER_SUPPLY_TECHNOLOGY_LIFE=4
uint8 POWER_SUPPLY_TECHNOLOGY_NICD=5
uint8 POWER_SUPPLY_TECHNOLOGY_LIMN=6
std_msgs/Header header
float32 voltage
float32 current
float32 charge
float32 capacity
float32 design_capacity
float32 percentage
uint8 power_supply_status
uint8 power_supply_health
uint8 power_supply_technology
bool present
float32[] cell_voltage
string location
string serial_number
```

Figure 5.3.5: *sec_msgs::BatteryState* structure

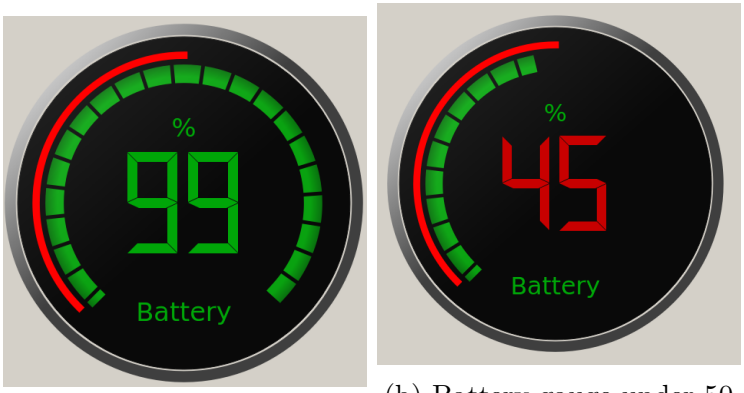
Figure 5.3.6a shows how is the gauge when the battery is above 50 % and Figure 5.3.6b shows how is the gauge when the battery is below 50 %, the number turn on red to warm the driver that has to be worried about the battery.

5.3.1.3 Lights state

The developed vehicle has different lights and the driver has to know the state of each them.

Lights are controlled by an Arduino that send the topic by a Raspberry Pi, the topic name is */car_state/lights* as a *sec_msgs::Lights* which structure can be seen in Figure 5.3.7.

Lights that are implemented on the vehicle are low beam and high beam, which allow the driver to see the roadway at night. Tail lights helps the drivers who are travelling



(a) Battery gauge.

(b) Battery gauge under 50 %.

Figure 5.3.6: Battery gauge displayed on the information HMI.

```
Header header
bool position
bool low_beam
bool high_beam
bool reverse
bool brake
bool left_turn_indicator
bool right_turn_indicator
bool emergency
bool all_lights
```

Figure 5.3.7: *sec_msgs::Lights* structure

behind you to recognize you and how far ahead you are. Signal lights, also known as turn signals or "blinkers", indicate to other drivers that you will soon be turning. Brake lights indicate that your vehicle is slowing down or stopping. Hazard lights warm other drivers that you are experiencing a problem, are in distress, or warning of an immediate danger.

Figure 5.3.8 shows the lights that can be seen on the HMI when they are on.

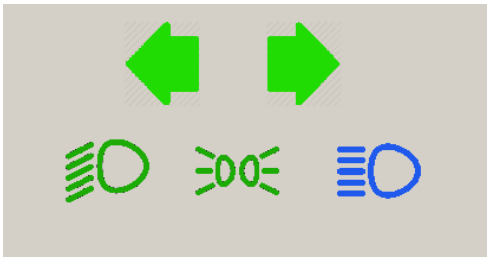


Figure 5.3.8: Lights icon on the HMI

5.3.1.4 Car state

The HMI will show which is the current driving mode that lead the vehicle between manual which allows the driver to have full control of the car, and autonomous, where the control is taken by the navigation system. In this mode the driver can take the control

by only using the throttle, brake or steering wheel.

Topic name is */car_state/CarState* as a *sec_msgs::CarState* which structure can be seen in Figure 5.3.9. Figure 5.3.10 shows how the HMI indicates that the car is on autonomous mode.

```
Header header
bool manual_mode
bool ready_mode
bool automatic_mode
bool actuated_brake
bool actuated_wheel
bool reverse
bool forward
bool parked
```

Figure 5.3.9: *sec_msgs::CarState* structure.

Figure 5.3.10: Autonomous mode warm on the HMI.

Because an electric car does not need a clutch, it also does not require gears. Electric vehicles do not feature a multi-speed gearbox like conventional petrol or diesel vehicles. Instead, they have just one gear. So the HMI will show how is the state of this gear, if it is set on direct, reverse or parking as shows in Figure 5.3.11.

Figure 5.3.11: Gear state, bold letter means the gear select, between Parking (P), Reverse (R) and Direct (D).

5.3.1.5 Localization mode

Techs4Age car has a D-GPS which gives differential corrections to the position but this advanced GPS depends on various factors that change along the drive. This is very important, specially when the vehicle is in autonomous mode, because it will only work fine if the car has differential corrections. So, in order to show the driver if the D-GPS is working correctly, a graphic was design to show him how is the performance of this system and if he can change to the autonomous mode.

Localization topic called */localization/fix* has a *sensor_msg::savSatFix* structure as shown in Figure 5.3.12. From this topic the HMi will extract and show as a visual design the number of satellites that are tracking the D-GPS and if it has differential corrections, which gives a better position result.

```
uint8 COVARIANCE_TYPE_UNKNOWN=0
uint8 COVARIANCE_TYPE_APPROXIMATED=1
uint8 COVARIANCE_TYPE_DIAGONAL_KNOWN=2
uint8 COVARIANCE_TYPE_KNOWN=3
std_msgs/Header header
sensor_msgs/NavSatStatus status
float64 latitude
float64 longitude
float64 altitude
float64[9] position_covariance
uint8 position_covariance_type
```

Figure 5.3.12: *sensor_msg::savSatFix* structure.

Figure 5.3.13 shows the three different states that the differential corrections can have.

(a) Real Time Kinematic. (b) Floating RTK. (c) GPS.

Figure 5.3.13: D-GPS

5.3.1.6 Perception layer

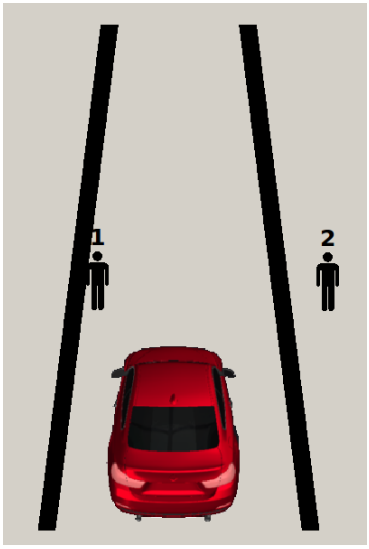
Understanding the environment is a crucial task to the development of an autonomous vehicle and this information can be useful for the driver when the vehicle is in manual mode. Because of this, the HMI has implemented a perception system that shows the pedestrians in front of the car detected by the perception layer.

To achieve this task, a Deep-Learning based Multi-Object Tracking has been implemented in the car, which use the ZED camera installed on the rack. This system is based on CenterNet approach [75] which is a CNN that detects each object (pedestrian, car, bike) as a triplet (top-left corner, center estimation and bottom-right corner), rather than a pair (only the corners) of keypoints, which improves both precision and recall. And

Simple Online and Real Time (SORT) [76] tracking is applied as well because it is a pragmatic approach to MOT (Multi-Object Tracking). SORT performs Kalman filtering in image space and frame-by-frame data association using the Hungarian method [77] with an association metric that measures bounding box overlapping.

Perception is published in the ROS ecosystem as a topic called */perception/list_object_tracking* which has a *yolo_ros::yolo_list::ConstPtr* structure. The HMI shows the pedestrian and car in front of the car, showing also the identifier assigned by the tracking system.

Figure 5.3.14 shows how this work on the CARLA simulator with two pedestrian in front of the car and how they are represented on the HMI. Figure 5.3.15 shows how is the HMI performance with one pedestrian in a real environment.

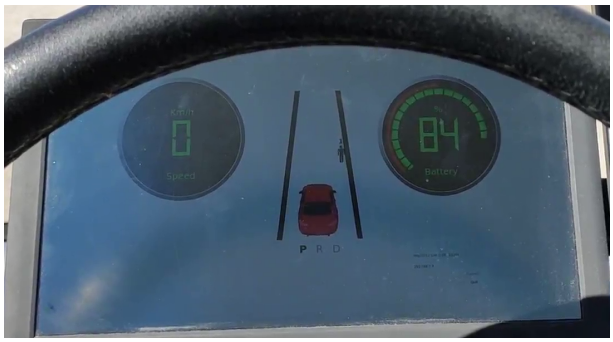


(a) HMI look of the perception module installed on.

(b) Car view of the two pedestrian with their bounding boxes.

(c) External look of the scene.

Figure 5.3.14: Carla simulation of the perception module of the HMI.



(a) HMI look of the perception module installed on.

(b) Car view of the pedestrian with his bounding box.

(c) External look of the scene.

Figure 5.3.15: Carla simulation of the perception module of the HMI.

5.3.2 Control HMI

The second display on the car is a touch screen which allows the driver to control some functions with his hand. This screen is connected to the computer that manage the control system as shows in Figure 5.3.16.

This system is based on a laptop with high performance but it will be replaced in the near future by a NVIDIA Jetson that has lower energy consumption and is prepared to do the required task as soon as the control swap from the laptop to this embedded system.

The driver has access to this screen to select the autonomous path and begin the travel. This layer of the HMI is very simple but will have more functions on the near future when the system who power on the screen will be a NVIDIA Jetson.

Figure 5.3.16: User launching the path using the touch screen on the autonomous vehicle called Tech4AgeCar.

Chapter 6

Results

This chapter presents the results obtained using both gaze tracking calibration method for building visual attention model. The first subsection will show the results by the linear calibration and the second one the obtained by the NARMAX calibration method.

6.1 Gaze focalization linear calibration

This section presents experimental results obtained for our architecture proposal using the linear calibration method. Evaluation is divided in two different subsections, firstly a precision test done to test the accuracy of the tool and secondly a comparison with other state-of-the-art system in a challenging dataset based on accidents, which is the main problem of distraction while driving, is carried out.

6.1.1 Visual attention model evaluation

To evaluate the precision of our driver attention model we have applied the following testing procedure. After the calibration step, the user has to look at eight points, placed on different positions on the screen, during eight seconds for each one. To get a more stable view, the first second of each point is despised. This procedure has been done to test the camera parameters, the camera position, the loss of precision by using glasses and finally the performance with different users. The metric used to evaluate the accuracy of the tool is the root mean square error (RMSE) of the gaze projection on the screen and its ground-truth. Accuracy is shown in total and regarding the (X,Y) axis and is evaluated in percentage, pixels and millimetres.

6.1.2 Camera parameters.

To reach optimum results, a previous study about camera parameters was carried out. In this aspect, resolution and camera frame rate were changed to test precision for the different options. ZED camera allows four different resolutions with different frame rates, as we show on Table 6.1. Performance shows the ability of OpenFace to process frames. This test was done only by one user, but to achieve more robust conclusions each test was repeated 5 times.

Results show that, on the X axis, best performance is get by VGA resolution at 100 frames per seconds. On the Y axis, best performance is get by HD720. In total, best accuracy is obtained for HD720. In conclusion, this last configuration is taken for the following tests. Also performance shows that OpenFace is not able to work at 100 Hz with an VGA resolution.

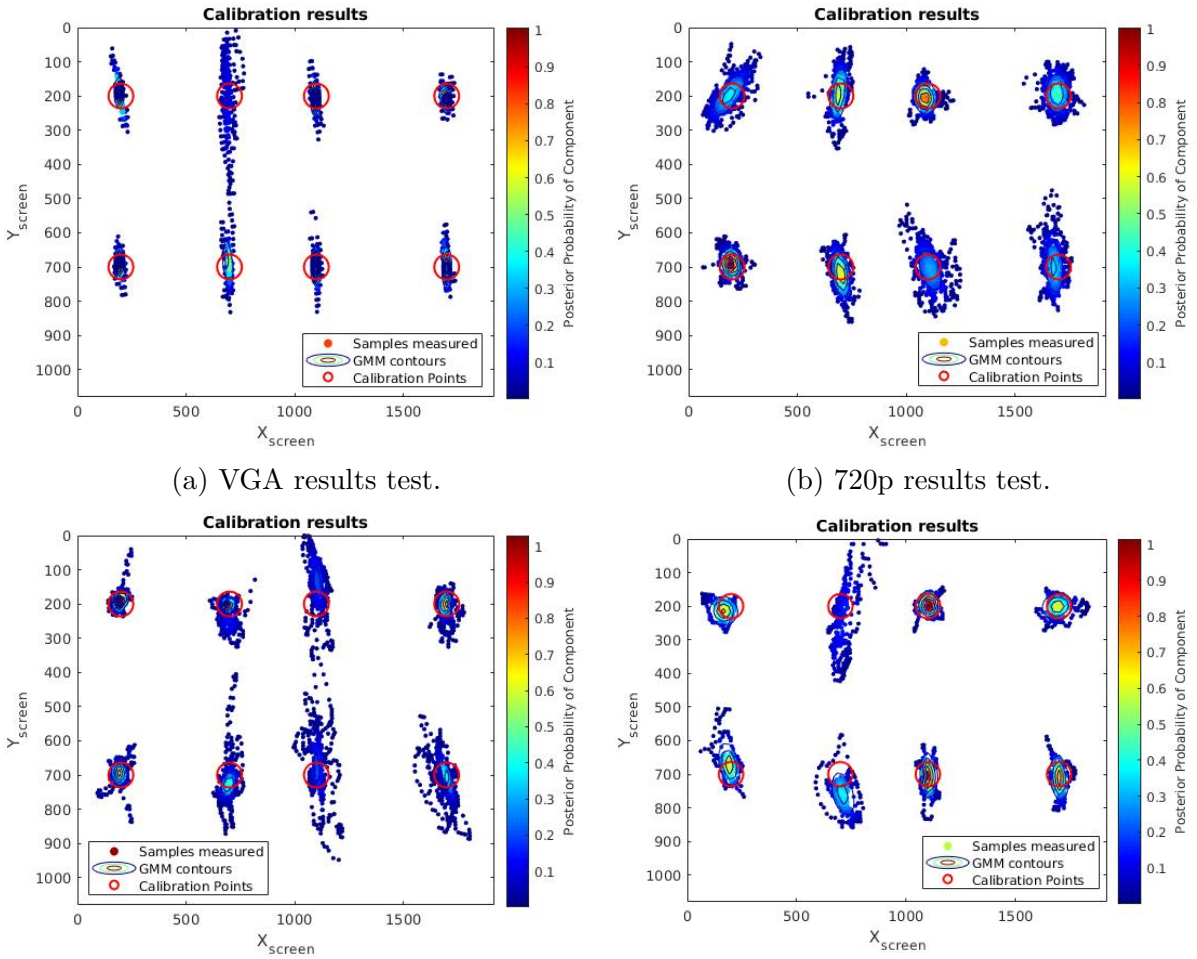
Figure 6.1.1 shows the obtained results by the four different camera configurations. Red circles indicate the goal points where the person has to look at. The heat map represents the spatial distribution of the data, divided in this case in eight different clusters, modelled by a Gaussian Mixture Model (GMM) of eight Gaussian. The heat map for each Gaussian indicate the probability of a pixel to belong to each cluster. In theory, the centre of each Gaussian should be placed over the corresponding calibration point and its covariance indicates the measurement uncertainty. This will be the representation for all the precision test carried out on this work.

Table 6.1: Testing accuracy vs camera parameters on a 1920x1080 screen. Test done by one person looking at eight points on a screen. Test was done 5 times.

Resolution	Frame rate	Acc_x			Acc_y			Acc_total			Performance (Hz)
		%	pix	mm	%	pix	mm	%	pix	mm	
HD2K	15	1,9	36,5	11,4	5,4	57,9	18,1	4,0	77,2	24,1	14,761
HD1080	30	1,4	27,5	8,6	5,7	61,0	19,1	4,1	79,2	24,7	29,378
HD720	60	1,9	36,1	11,3	**4,0**	43,7	13,6	**3,2**	60,5	18,9	50,74
VGA	100	**0,6**	12,3	3,8	4,7	50,2	15,7	3,3	63,7	19,9	74,63

6.1.3 Camera position.

Camera position is vital for the system to work. Three different positions compatible with a real car were tested, all of them aligned with the middle of the screen but at different heights (10, 48 and 78 centimetres over the table where the screen is placed). Results are shown on Table 6.2 for the same conditions that the explained in the before section. The optimal position for the camera is just in front of the user, but this is not possible in a real car because hides the road scene. Top position is not allowed because face landmarks are



(a) VGA results test.

(b) 720p results test.

(c) 1080p results test.

(d) 2k results test.

Figure 6.1.1: Camera parameters results using lineal calibration. 8 points are displayed on the screen (Red circle). The results from our output creates 8 Gaussian around the testing points.

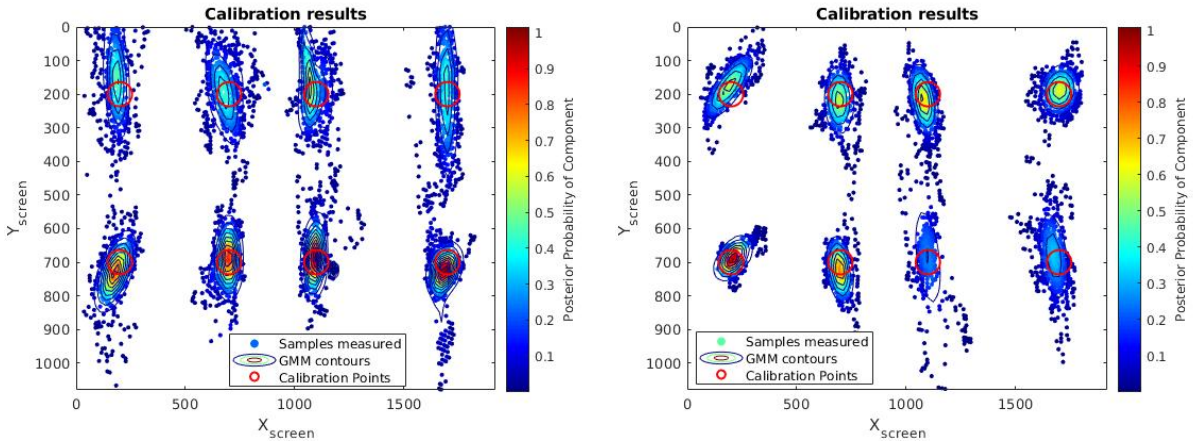
occluded most of the time and the model does not work properly. Then, bottom position was chosen.

Figure 6.1.2 shows the results obtained by the two different camera positions tested.

6.1.4 Glasses

Some drivers wear glasses while driving and some other drivers prefer to drive with sunglasses. Performance of every gaze monitoring system must be analysed under these common situations. Commercial applications like Tobii Pro Glasses [78] do not work properly because their sensors are on some special glasses and two different glasses cannot be used at the same time.

In our case, precision will get worse for sure due to the glasses hide some face landmarks. To evaluate impaired performance three different test were done: no glasses, with



(a) Bottom position results test.

(b) Medium position results test.

Figure 6.1.2: Camera position results using lineal calibration. 8 points are displayed on the screen (Red circle). The results from our output creates 8 Gaussian around the testing points.

Table 6.2: Testing accuracy vs camera position on a 1920x1080 screen. Test done by one person looking at eight points on a screen. Test was done 5 times.

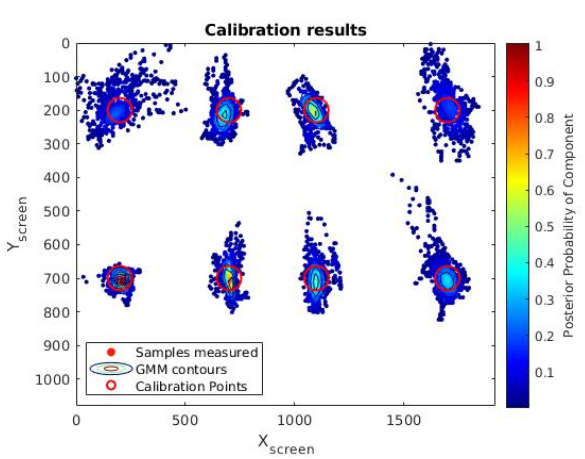
Position	Acc_x			Acc_y			Acc_total		
	%	pix	mm	%	pix	mm	%	pix	mm
Top	-	-	-	-	-	-	-	-	-
Mid	**1,9**	36,1	11,3	**4,0**	43,7	13,6	**3,2**	60,5	18,9
Bot	**2,5**	48,0	15,0	**10,6**	114,5	35,8	**7,7**	147,9	46,2

glasses and with sun glasses, in the same conditions that in the before case. Results are depicted on Table 6.3. For the sunglasses case, our system does not work because eyes landmarks are hidden and gaze can not be calculated. For the glasses case, accurate numbers decrease a little but can be used without problems.

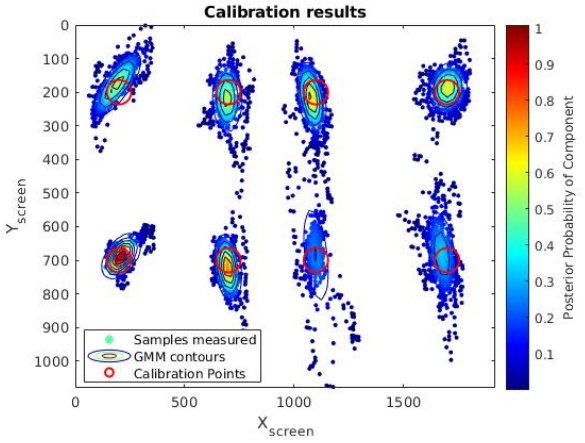
Figure 6.1.3 shows the results obtained by a user using glasses and the same user without using them.

Table 6.3: Testing accuracy vs glasses on a 1920x1080 screen. Test done by one person looking at eight points on a screen. Test was done 5 times

Eyes-glasses	Acc_x			Acc_y			Acc_total		
	%	pix	mm	%	pix	mm	%	pix	mm
Free	**1,9**	36,1	11,3	**4,0**	43,7	13,6	**3,2**	60,5	18,9
Glasses	**2,5**	48,7	15,2	**4,6**	50,0	15,6	**3,7**	71,7	22,4
Sunglasses	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**



(a) Medium position results test with glasses.



(b) Medium position results test without glasses.

Figure 6.1.3: User glasses test results using lineal calibration. 8 points are displayed on the screen (Red circle). The results from our output creates 8 Gaussian around the testing points.

6.1.5 Precision test

Once the best camera parameters and camera position have been found, it is time to test the accuracy of our model with different users. 25 users were requested for the tests. Firstly, they were asked to calibrate the tool looking at 4 points as shown on Figure 6.1.4. Age of the users were between 17 and 55 years old and 3 of them wore glasses during the test.

Results for all of the users can be seen on Figure 6.1.5.

Our method achieves about 1.9 % error on the X axis and 4 % error on Y axis, being bellow the reported by OpenFace for the MPIIGaze dataset (9.1 degrees equivalent in our case to 16.5 % on X axis and 21 % on Y axis) [55]. This comparison should be taken with caution since in our case gaze movements are limited and the number of users who undergo the test is less.

Table 6.4: Testing accuracy on a 1920x1080 screen. Test done by one person looking at eight points on a screen. Test done by 25 users looking at eight points on a screen.

	Acc_x			Acc_y			Acc_total		
	%	pix	mm	%	pix	mm	%	pix	mm
25 people	**1,9**	36,1	11,3	**4,0**	43,7	13,6	**3,2**	60,5	18,9

Results on Y axis are worse due to the number of pixels for the vertical field of view range is smaller that the corresponding to the horizontal one. Despite these errors are considerable, they can be enough for our application. Figure 6.1.8 shows the error overprinted on an image with a Crash Object in order to compare scales. As we can see,

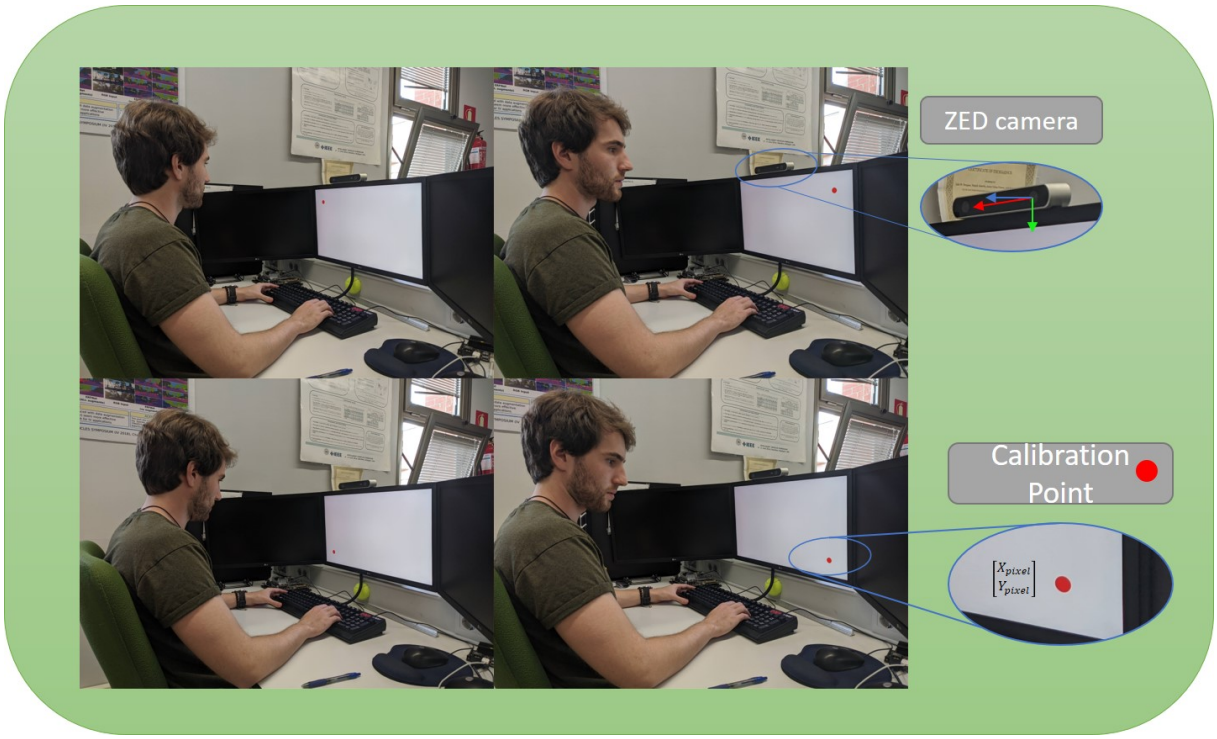


Figure 6.1.4: Calibration step. The user under test has to watch during 8 seconds 4 points displayed on the screen

in spite of the measurement uncertainty the object is perfectly detected in the frame, which demonstrate our method can be used to focalize accidents in complex scenarios.

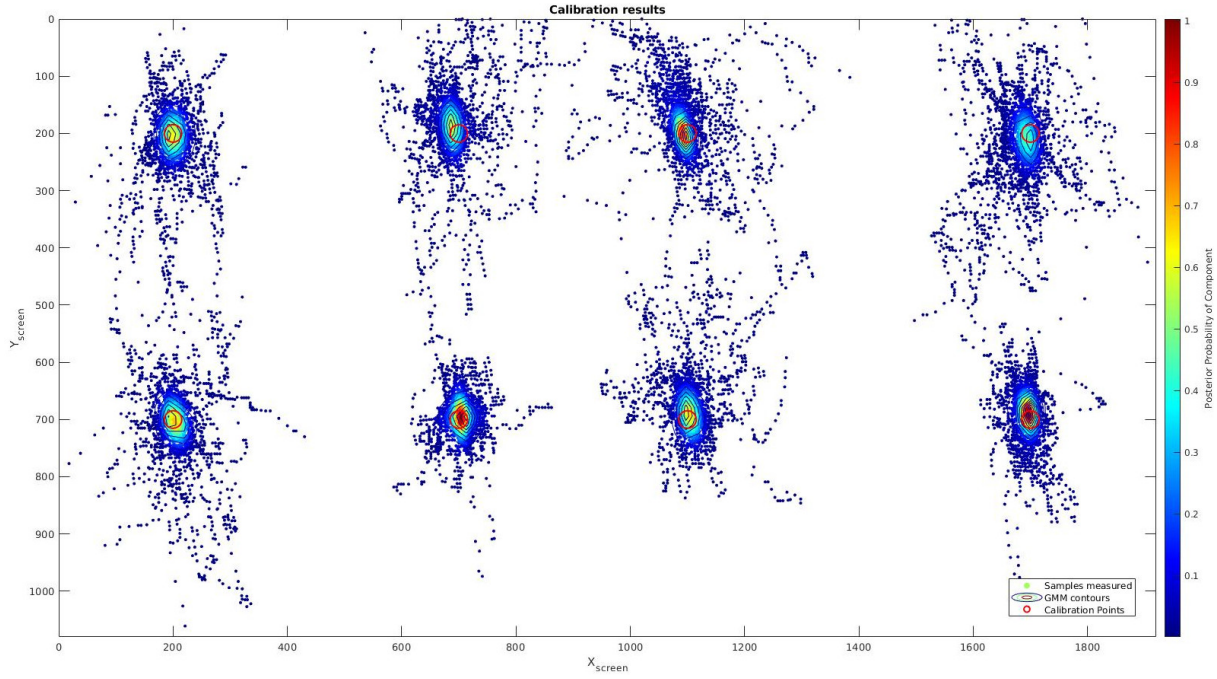


Figure 6.1.5: Calibration results. 8 points are displayed on the screen (Red circle). The results from our output creates 8 Gaussian around the testing points.

6.1.6 DADA2000 evaluation

Once shown the potential of our visual attention method to focalize objects of a certain size in an image, we are ready to test it on a challenging video benchmark with driver attention and driving accidents annotated to evaluate the performance of our proposal to estimate accidents in video sequences. To have a clear idea of the quality of our method we will compare results with the obtained using an expensive and active desktop-mounted eye-tracker watching the same videos in similar conditions. To validate it the application should reach an optimal result on a dataset obtained using an eye-tracker. Dataset based on saliency maps or computational models are not the correct way to test the tool because we think that Ground Truth provided by these kind of dataset are not the optimal to be compared with our proposal because are based on computational models and not in human gaze.

Different options were raised, such as MIT300 [79], CAT2000 [80], DR(eye)VE [20] and DADA2000 [2]. The two first ones were discarded because they are not focused on drivers. Between the two lasts, authors decided to use DADA2000 because it is focused on real traffic accidents, which are the critical moments to evaluate the driving attention.

DADA2000 was done collecting accident videos from different websites, and it is composed of 658,476 frames in 2000 videos with a resolution of 1584x660 pixels. At the time of this work, only half of data is public. Then, we use 1000 videos for the experiments, which are classified in three groups: training (598 videos), validation (198 videos) and testing (222 videos). The videos are divided into 54 accident categories classified into two large groups, ego-car involved and ego-car uninvolved as shows Figure 6.1.6.

Videos are recorded in different environment such as illumination conditions (day and night), weather (sunny, rainy and snowy) and occasion (urban, rural, highway and tunnel). The dataset provides the following information: annotated crash-objects position in pixels per frame, attention map for each frame and temporal window for each accident indicating the involved frames. Videos are partitioned in three main clips: before the accident, accident and after the accident, as it is depicted in Figure 6.1.7. Also, the accident clip is divided in three sub-sections for a better analysis (Start, Mid and End).

6.1.6.1 Crash Object detection in DADA2000 benchmark

To test Crash Object detection, we employed 4 volunteers who were invited to watch the testing section of the dataset, composed by 222 videos. For each visualization, the output of our visual attention model was recorded in a synchronized way with the video. Between videos, users experimented a pause of 3 seconds to accommodate the gaze for the next video. In order to wide the working field of view, dataset was resized from 1584x660 to 1920x1080 to match it with the screen resolution used in the experiment. The clips were

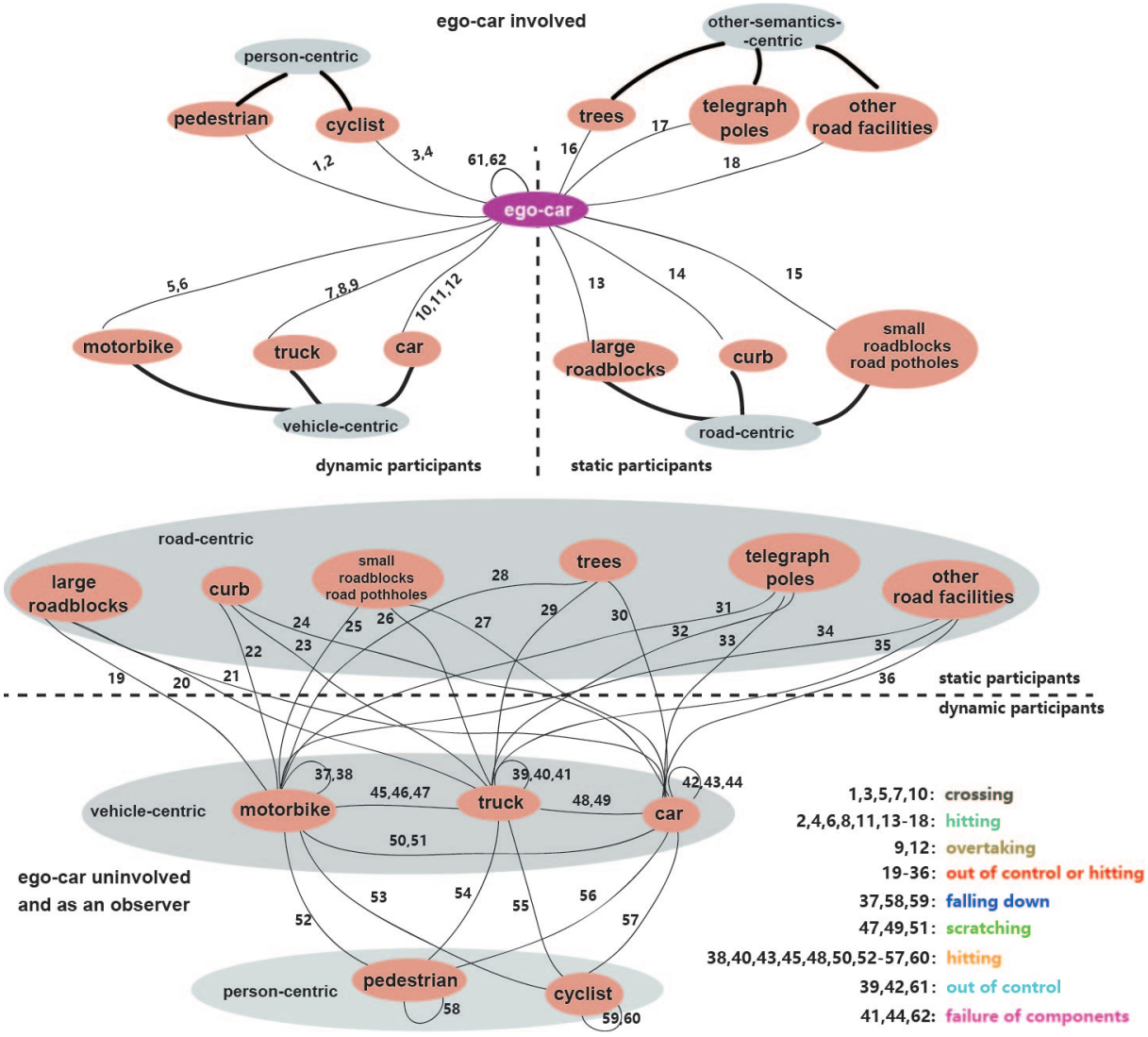


Figure 6.1.6: DADA 2000 accident classification of the 54 accident categories.

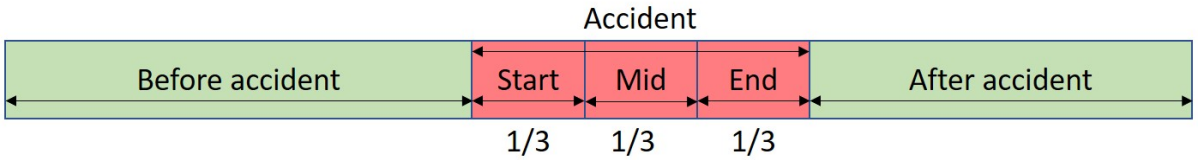


Figure 6.1.7: Temporal partition of the video. Dividing the accident part in three pieces of same length to evaluate the results by sections.

played at 30 fps to compare our results with the obtained by the authors of DADA2000 [2] in the same terms. We recorded attention map without temporal aggregation, obtaining two measures per frame, because our system run at 60 Hz.

We do not want to focus this experiment in which kind of accidents the users focus their attention in a better way, we want to test the possibility of using this kind of method to acquire driver attention. The goal of this test is to localize driver attention on a screen. We use the same metrics as the authors of the dataset to evaluate this

Figure 6.1.8: DADA2000 frame. Crash object circle for 60 and 300 pixels (blue). Error area (red).

parameter. We measure the distance between the Crash Object center (ground-truth manually annotated) and the attention points obtained for each frame. If this distance is below a certain threshold (Th) in pixels the detection is consider as true and false if it is higher. Figure 6.1.8 shows a frame with the Crash Object an its true detection area, represented by a circle, for a Th equals to 60 and 300 pixels. Performance detection is solved as a binary problem: True, if the attention point falls inside the circle, and False, if it falls outside. For this work, we only have tested experiments with the priori information option (with-priori), that means users are told that they have to find crash-objects in the sequences. With these premises we obtain two indicators:

- Frame ratio: percentage of frames in each clip section where the attention point is inside the correct detection area. This is measured frame by frame because the crash object change with it.
- Success rate: percentage of clips which frame ratio for the entire clip is over 50%. Clips refers different video sub-sections defined for the accident.

Results for different detection area sizes and a comparison with the numbers presented by the DADA2000 authors using a Senso Motoric Instrument (SMI) RED250 desktop-mounted infrared eye tracker (Figure 6.1.9) are shown on Table 6.5. Our success rate is higher than the obtained by the DADA200 authors for all the partitions except for the start one with a threshold of 60 pixels. However, our numbers are obtained for 4 users per video instead of the 5 they argue to use, even though they also argue every video was watched by at least two people, indicating that not all of them were watched by the 5 users. For a 300 pixels threshold we have obtained more than 90 % of success rate for start and mid sections of the accident event. The end section always get less success rate because, at this part of the video, users normally fix their attention in other part of the

Table 6.5: Success rate: percentage of number of clips that frame ratio for the entire clip is more than half of the frames of the entire accident scene.

System	Ours (based on OpenFace using linear calibration)			DADA2000		
Th (pixels)	Start	Mid	End	Start	Mid	End
60	9,4%	18,8%	10,9%	10,8%	10,7%	5,1%
100	39,1%	48,4%	29,7%	34,4%	30,8%	22,6%
160	76,6%	78,1%	57,8%	59,0%	57,2%	49,1%
200	81,3%	87,5%	65,6%	68,0%	67,2%	60,1%
260	90,6%	90,6%	71,9%	76,6%	76,6%	71,5%
300	90,6%	93,8%	78,1%	80,0%	81,3%	76,6%
360	93,8%	95,3%	87,5%	84,0%	81,0%	82,0%
400	93,8%	96,9%	90,6%	86,0%	86,0%	84,0%
460	96,9%	96,9%	95,3%	90,0%	88,0%	88,0%
500	98,4%	96,9%	96,9%	91,0%	90,0%	90,0%
560	98,4%	98,4%	96,9%	93,0%	92,0%	92,0%
600	98,4%	98,4%	96,9%	94,0%	95,0%	93,0%
660	98,4%	98,4%	96,9%	95,0%	96,0%	96,0%
700	98,4%	98,4%	96,9%	95,0%	97,0%	96,0%
760	98,4%	98,4%	96,9%	96,0%	97,0%	96,0%
800	98,4%	98,4%	98,4%	97,0%	98,0%	96,0%
860	98,4%	98,4%	100,0%	98,0%	99,0%	97,0%
900	100%	98,4%	100%	98,0%	99,0%	97,0%
960	100%	98,4%	100%	98,0%	100%	98,0%
1000	100%	98,4%	100%	99,0%	100%	98,0%

Table 6.6: Frame ratio: percentage of frames in each clip section where the attention point is inside the correct detection area. This is measured frame by frame because the crash object change with it.

System	Ours (based on OpenFace using linear calibration)			DADA2000		
Th (pixels)	Start	Mid	End	Start	Mid	End
60	19,5%	23,7%	17,3%	17,0%	17,0%	13,0%
100	37,4%	44,6%	32,8%	25,0%	36,0%	30,0%
160	64,0%	64,0%	64,0%	58,0%	57,0%	50,0%
200	73,7%	81,2%	64,7%	67,0%	67,0%	60,0%
260	83,1%	89,5%	76,6%	75,0%	77,0%	68,0%
300	86,6%	92,4%	81,2%	78,0%	83,0%	75,0%
360	90,1%	95,3%	87,7%	84,0%	85,0%	80,0%
400	91,2%	96,3%	91,1%	86,0%	87,0%	82,5%
460	93,3%	97,3%	93,9%	87,0%	92,0%	85,0%
500	94,0%	97,8%	95,2%	91,0%	92,0%	87,0%
560	95,0%	98,0%	96,4%	92,0%	94,0%	90,0%
600	95,8%	98,0%	97,3%	94,0%	95,0%	92,5%
660	96,9%	98,3%	98,3%	95,0%	97,0%	95,0%
700	97,6%	98,3%	98,5%	96,0%	97,0%	96,0%
760	98,4%	98,4%	98,9%	96,0%	98,0%	97,0%
800	98,7%	98,5%	99,1%	97,0%	98,0%	97,0%
860	98,9%	98,9%	99,2%	97,0%	98,0%	97,0%
900	99,1%	99,1%	99,3%	98,0%	99,0%	98,0%
960	99,6%	99,4%	99,4%	99,0%	99,0%	98,0%
1000	99,7%	99,4%	99,5%	100%	100%	98,0%

scene. Results are or par or even better than the claimed by the authors of DADA2000. Results shows that early accident detection is worse than for the middle partition because users take some time to react. In the last part of the accident some users under test keep their eyes out the accident and the late accident detection gets worse. Table 6.6 shows the Frame ratio obtained with the method proposed.

Figure 6.1.9: Senso Motoric Instrument (SMI) RED250 desktop-mounted infrared eye tracker

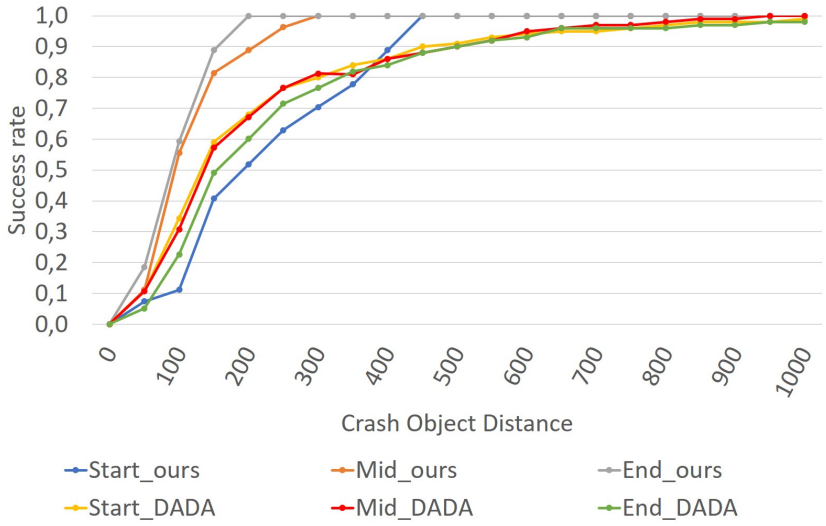
Figure 6.1.10a shows the success rate and frame ratio curves, obtained for a threshold between 60 and 1000 pixels and for the three sub-sections presented in the accident partition, in the same way that the shown by the dataset authors in [2]. Our results outperform the obtained for our baseline for all the analysed sub-sections.

6.1.6.2 Heat attention map on DADA2000

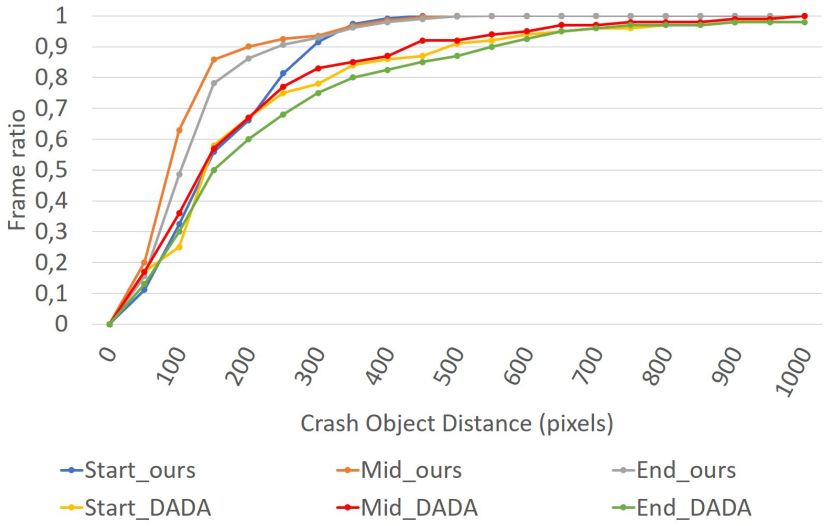
Heat attention map is a good visualization tool to represent where a user is looking when he is driving. To represent the trajectory of the gaze, a temporal window is used. For the presented experiments the temporal window has been set to 1 second, like in the DR(eye)VE project, which corresponds to 60 measures.

Figure 6.1.11 shows the way this experiment has been done. The user is looking at the accident while the camera is recording him. The attention map generated by the samples is shown. As we can see, our attention map is included in the baseline map, showing the usability of our tool but using a cheaper and non-intrusive sensor.

Figure 6.1.12 shows how temporal aggregation on heat attention map can help to predict where the accident will happen. The figure shows 2 accident situations with their



(a) Success rate obtained by the system proposed.



(b) Frame ratio obtained by the system proposed.

Figure 6.1.10: Results obtained on the testing

corresponding maps in the image where the accident takes place as a heat map. Images displayed are with a temporal difference of 1 second. As we can see, the heat map is able to correctly predict all the crash objects. The figure shows the attention map generated by 4 users.

6.2 Gaze focalization NARMAX calibration

This section presents experimental results obtained for our architecture proposal using the non-linear calibration method called NARMAX. Evaluation is divided in two different subsections, firstly a precision test is done to test the accuracy of the tool and secondly a comparison with other state-of-the-art system in the same challenging dataset that for

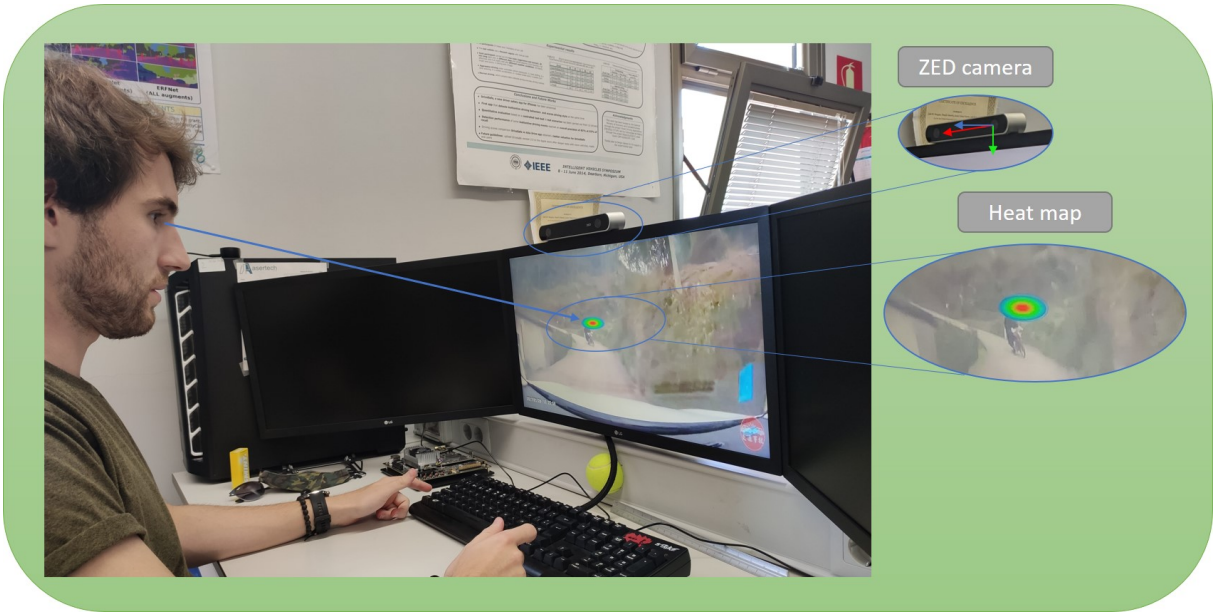


Figure 6.1.11: Heat attention map captured on real time during a DADA test.

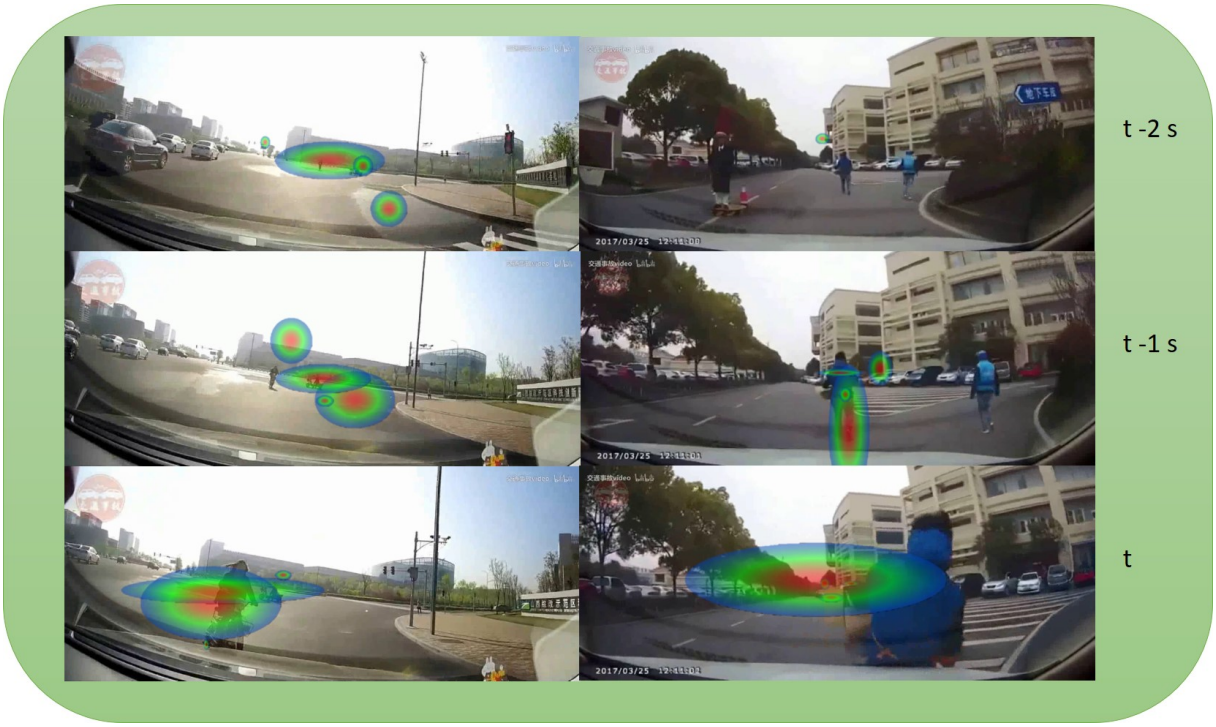


Figure 6.1.12: Attention map with temporal aggregation for the frame t, t-1 s and t-2 s.

the linear case is carried out.

6.2.1 Visual attention model evaluation

To evaluate the precision of our driver attention model we have applied the same testing procedure as done previously with the linear calibration. After the calibration and training step, the user has to look at eight points, placed on different positions on the screen, during

eight seconds for each one. To get a more stable view, the first second of each point is despised. This procedure has been done to test the camera parameters, the camera position, the loss of precision by using glasses and finally the performance with different users. The metric used to evaluate the accuracy of the tool is the root mean square error (RMSE) of the gaze projection on the screen and its ground-truth. Accuracy is shown in total and regarding the (X,Y) axis and is evaluated in percentage, pixels and millimetres.

6.2.2 Camera parameters.

As done with the linear calibration, a first study about best camera parameters is needed. In this aspect, resolution and camera frame rate were changed to test precision for the different options. ZED camera allows four different resolutions with different frame rates as show on Table 6.7. Performance shows the ability of OpenFace to process frames. This test was done only by one user, but to achieve more robust conclusions each test was repeated 5 times.

Results show that, on the X axis, the best performance is get by 2K resolution at 15 frames per second. On the Y axis, best performance is get by HD1080. In total, best accuracy is obtained for HD1080 which is difference from the linear calibration results showed early, which best camera parameters were 720p at 60 frames per second. In conclusion, HD1080 at 30 frames per second are the camera parameters used for the next tests performed with the NARMAX calibration.

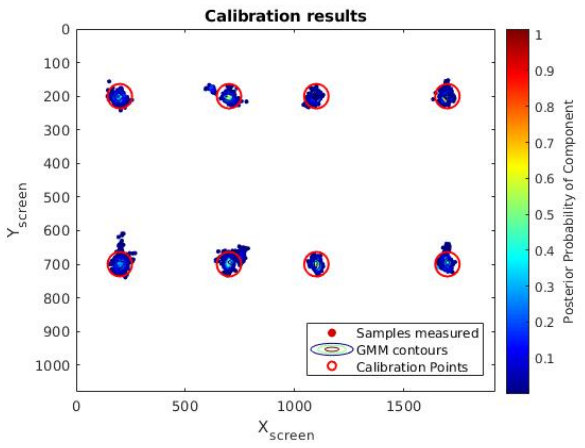
Figure 6.2.1 shows the results obtained by the four different camera configurations. This test has been done 5 times for each camera configuration. Results shows a quite improvement from linear calibration method which obtain an error of 60.5 pixels, which is 3.5 times higher than the obtained with this method.

Table 6.7: Testing accuracy vs camera parameters on a 1920x1080 screen using NARMAX calibration method. Test done by one person looking at eight points on a screen. Test was done 5 times.

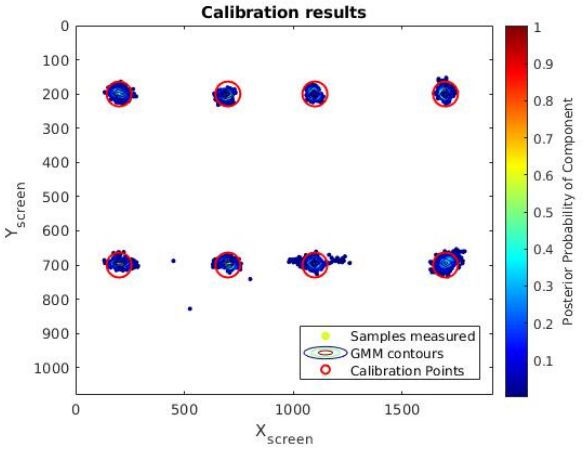
Resolution	Frame rate	Acc_x			Acc_y			Acc_total			Performance (Hz)
		%	pix	mm	%	pix	mm	%	pix	mm	
HD2K	15	**0,7**	12,5	3,9	1,3	14,5	4,5	1,1	20,3	6,3	14,8
HD1080	30	0,7	12,6	3,9	**1,1**	11,8	3,7	**0,9**	17,3	5,4	29,47
HD720	60	1,6	29,9	9,3	0,9	10,2	3,2	1,3	24,7	7,7	53,447
VGA	100	0,9	18,2	5,7	1,1	12,4	3,9	1,1	20,2	6,3	76,16

6.2.3 Camera position.

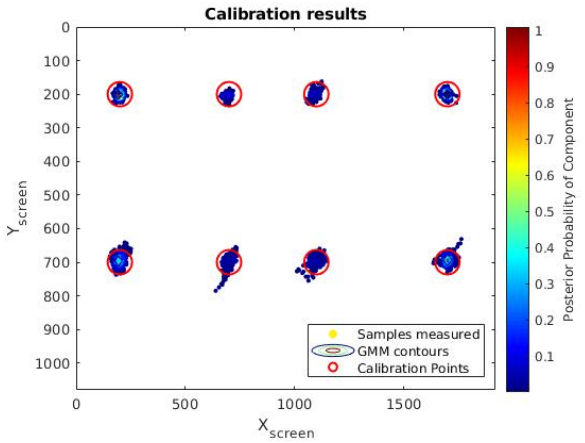
As done with the linear calibration camera position is vital for the system to work and have to be calidate if with this new calibration method the three position tested are



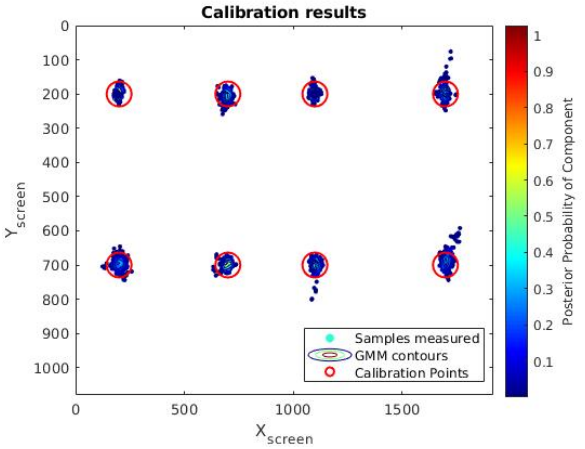
(a) VGA results test.



(b) 720p results test.



(c) 1080p results test.



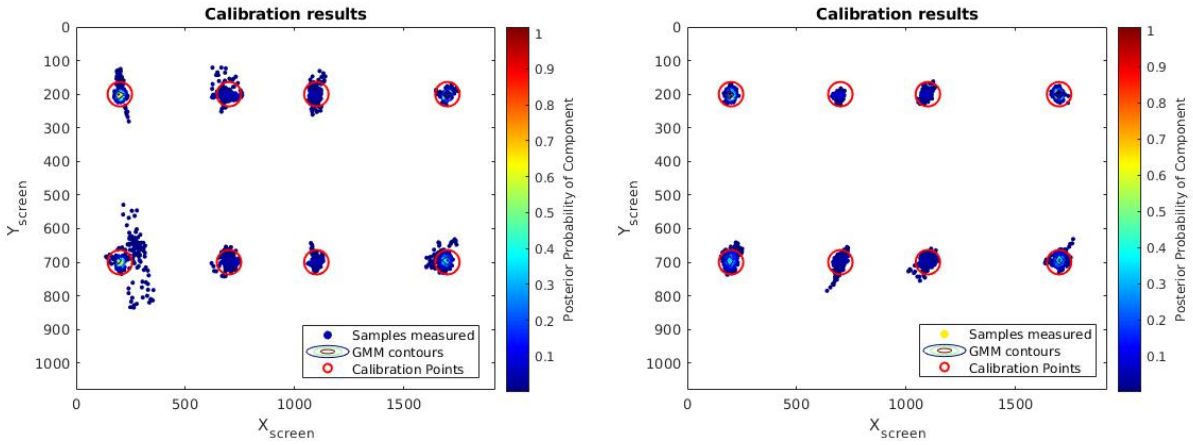
(d) 2k results test.

Figure 6.2.1: Camera parameters results NARMAX calibration. 8 points are displayed on the screen (Red circle). The results from our output creates 8 Gaussian around the testing points.

available.

Results are shown on Table 6.8 for the same conditions that the explained in the before section. The optimal position for the camera is just in front of the user, but this is not possible in a real car because hides the road scene. Top position is not allowed because face landmarks are occluded most of the time and the model does not work properly as done with the linear calibration, this results can not be changed with this new calibration method because OpenFace lose some landmarks and cannot find the face. Then, bottom position was chosen to place it on the car, as done with linear calibration.

Figure 6.2.2 shows the results obtained by the two different tested camera positions. Results obtained shows an improvement of more than 6 times for the bottom position and near 3.5 for the middle position.



(a) Bottom position results test.

(b) Medium position results test.

Figure 6.2.2: Camera position results using NARMAX calibration. 8 points are displayed on the screen (Red circle). The results from our output creates 8 Gaussian around the testing points.

Table 6.8: Testing accuracy vs camera position on a 1920x1080 screen. Test done by one person looking at eight points on a screen. Test was done 5 times.

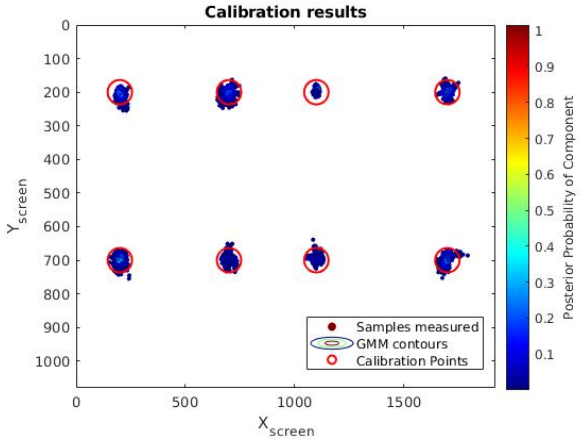
Position	Acc_x			Acc_y			Acc_total		
	%	pix	mm	%	pix	mm	%	pix	mm
Top	-	-	-	-	-	-	-	-	-
Mid	**0,7**	12,6	3,9	**1,1**	11,8	3,7	**0,9**	17,3	5,4
Bot	1,0	18,6	5,8	1,5	16,1	5,0	1,3	24,1	7,5

6.2.4 Glasses

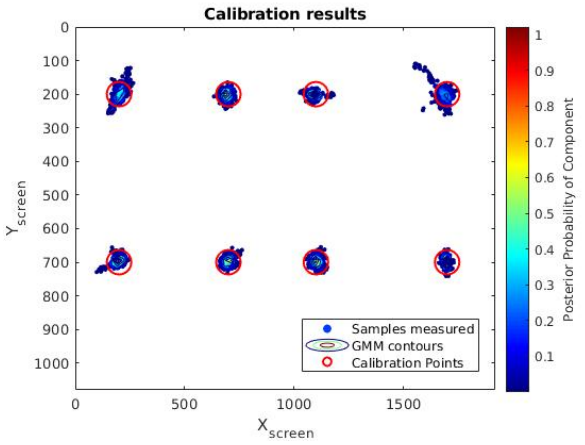
Some drivers wear glasses while driving and some other drivers prefer to drive with sun-glasses. Results obtained previously with the calibration linear method showed that it cannot work with an user wearing sun-glasses, which limits the use of the tool proposed.

With this calibration method, performance is even better using glasses than without them, this can be because user made some unusual movements during the test without them which will cause this difference. Even so, difference between them is less than a pixel which is an insignificant difference. Both test were done using the same lightning conditions which will differ in a real experiment where the sun may cause reflexes on the glasses. Results are displayed on Table 6.9. With this calibration method it is possible to track the gaze even with sunglasses, this is because this process take into account more parameters from the face than the gaze angle that will introduce error to the measure because of the partial occlusion of the eyes.

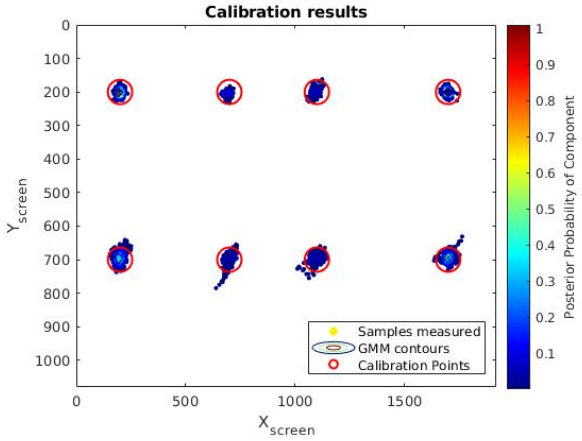
Figure 6.2.3a shows the results obtained by an user using glasses, figure 6.2.3b with sun glasses and figure 6.2.3c the same user without using them. Results show an improvement over the linear calibration between 3 and 4 times better.



(a) Medium position results test with glasses.



(b) Medium position results test with sunglasses.



(c) Medium position results test without glasses.

Figure 6.2.3: User glasses test results using NARMAX calibration. 8 points are displayed on the screen (Red circle). The results from our output creates 8 Gaussian around the testing points.

6.2.5 Precision test

Once the best camera parameters and camera position have been found with one user, it is time to test the accuracy of our model with different users. 10 users were requested for the tests. Firstly, they were asked to calibrate the tool. For this purpose they have to look at eight points during eight seconds each, this data will be send to the python script to calculate the parameters of the NARMAX model in order to calibrate the tool for each user.

Results for all of the users can be seen on Figure 6.2.4. Our method achieves about 1,02 % error on the X axis and 1,04 % error on Y axis, which is an error 3 times less than the obtained by the linear calibration. Unlike linear calibration method this system provides a similar accuracy in both axis which is an important improvement because it allows to have a circle attention map instead of a ellipse. Figure 6.2.5 shows the error

Table 6.9: Testing accuracy vs glasses on a 1920x1080 screen. Test done by one person looking at eight points on a screen. Test was done 5 times using NARMAX calibration

Eyes-glasses	Acc_x			Acc_y			Acc_total		
	%	pix	mm	%	pix	mm	%	pix	mm
Free	0,7	12,6	3,9	1,1	11,8	3,7	0,9	17,3	5,4
Glasses	**0,6**	11,6	3,6	**1,0**	11,3	3,5	**0,9**	16,4	5,1
Sunglasses	0,9	17,5	5,5	1,3	13,6	4,2	1,1	21,1	6,6

overprinted on an image with a Crash Object in order to compare scales. Also, the linear calibration error is displayed in order to check the difference between both methods. This new result allows to use a better tool in order to obtain the attention map generated by an user driving.

Table 6.10: Testing accuracy on a 1920x1080 screen. Test done by 10 users looking at eight points on a screen using the NARMAX calibration method.

	Acc_x			Acc_y			Acc_total		
	%	pix	mm	%	pix	mm	%	pix	mm
10 people	**1,02**	19,5	10,4	**1,04**	11,2	6,0	**1,03**	19,7	6,2

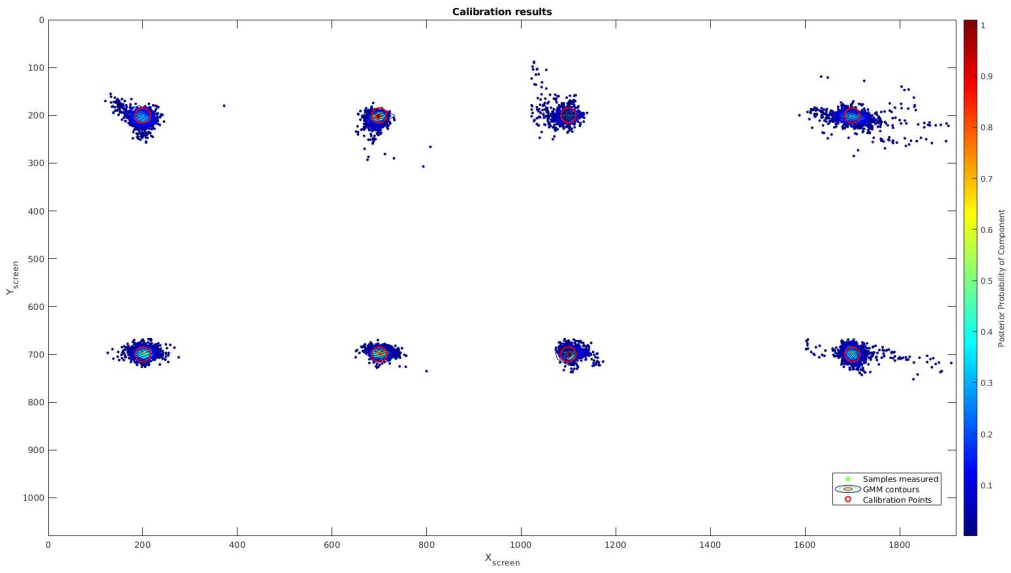


Figure 6.2.4: Calibration results using NARMAX calibration method. 8 points are displayed on the screen (Red circle). The results from our output creates 8 Gaussian around the testing points.

6.2.6 DADA2000 evaluation

Once shown the potential of our new visual attention method, based on NARMAX model, to focalize objects of a certain size in an image, we are ready to test it on a challenging

video benchmark with driver attention and driving accidents annotated to evaluate the performance of our proposal to estimate accidents in video sequences as done with the linear calibration.

To have a clear idea of the quality of our method we will compare results with the obtained using an expensive and active desktop-mounted eye-tracker watching the same videos in similar conditions in a same way as done with the linear calibration. This evaluation will be done using DADA2000 dataset, which is a dataset composed by accidents.

The dataset provides the following information: annotated crash-objects position in pixels per frame, attention map for each frame and temporal window for each accident indicating the involved frames. Videos are partitioned in three main clips: before the accident, accident and after the accident, as it is depicted in Figure 6.1.7. Also, the accident clip is divided in three sub-sections for a better analysis (Start, Mid and End).

6.2.6.1 Crash Object detection in DADA2000 benchmark

Crash Object detection has been tested with 4 users who were invited to watch the videos, in the same way as done using the linear calibration. Between videos, users experimented a pause of 3 seconds to accommodate the gaze for the next video. Dataset was resized from 1584x660 to 1920x1080 to match it with the screen resolution used in the experiment, in order to wide the working field of view.

The clips were played at 30 fps to compare our results with the obtained by the authors of DADA2000 [2] in the same terms. We recorded attention map without temporal aggregation, obtaining one measure per frame, because with this calibration method has been validated that the best camera parameters are HD1080 at 30 frames per second which will give one measure per frame.

This test is not focus on which kind of accidents the users focalize better. This test is done in order to validate the possibility of using this kind of method to acquire driver attention. The goal of this test is to localize driver attention on a screen. We use the same metrics as the authors of the dataset to evaluate this parameter. The distance between the Crash Object center (ground-truth manually annotated) and the attention point obtained for each frame is measured. If this distance is below a certain threshold (Th) in pixels the detection is consider as true and false if it is higher. With these premises we obtain two indicators (Frame ratio and Success rate).

Results are compared with the obtained by DADA2000 authors using a Senso Motoric Instrument (SMI) RED250 desktop-mounted infrared eye tracker, results shows different detection area sizes on Table 6.11. Our success rate is higher than the obtained by the DADA200 authors for all the partitions except for the start one with a threshold of 60 pixels. However, our numbers are obtained for 4 users per video instead of the 5 they

Table 6.11: Success rate: percentage of number of clips that frame ratio for the entire clip is more than half of the frames of the entire accident scene.

System	Ours (based on OpenFace using NARMAX calibration)			DADA2000		
Th (pixels)	Start	Mid	End	Start	Mid	End
60	20,3%	53,1%	64,1%	10,8%	10,7%	5,1%
100	32,8%	75,0%	84,4%	34,4%	30,8%	22,6%
160	40,7%	85,9%	96,3%	59,0%	57,2%	49,1%
200	54,7%	90,6%	96,9%	68,0%	67,2%	60,1%
260	65,6%	92,2%	96,9%	76,6%	76,6%	71,5%
300	70,3%	95,3%	96,9%	80,0%	81,3%	76,6%
360	87,5%	98,4%	98,4%	84,0%	81,0%	82,0%
400	92,2%	98,4%	98,4%	86,0%	86,0%	84,0%
460	95,3%	98,4%	100%	90,0%	88,0%	88,0%
500	98,4%	98,4%	100%	91,0%	90,0%	90,0%
560	98,4%	100%	100%	93,0%	92,0%	92,0%
600	100%	100%	100%	94,0%	95,0%	93,0%
660	100%	100%	100%	95,0%	96,0%	96,0%
700	100%	100%	100%	95,0%	97,0%	96,0%
760	100%	100%	100%	96,0%	97,0%	96,0%
800	100%	100%	100%	97,0%	98,0%	96,0%
860	100%	100%	100%	98,0%	99,0%	97,0%
900	100%	100%	100%	98,0%	99,0%	97,0%
960	100%	100%	100%	98,0%	100%	98,0%
1000	100%	100%	100%	99,0%	100%	98,0%

Table 6.12: Frame ratio: percentage of frames in each clip section where the attention point is inside the correct detection area. This is measured frame by frame because the crash object change with it.

System	Ours (based on OpenFace using NARMAX calibration)			DADA2000		
Th (pixels)	Start	Mid	End	Start	Mid	End
60	48,0%	64,3%	62,9%	17,0%	17,0%	13,0%
100	63,2%	82,0%	82,5%	25,0%	36,0%	30,0%
160	63,4%	90,9%	89,1%	58,0%	57,0%	50,0%
200	85,2%	93,6%	94,0%	67,0%	67,0%	60,0%
260	91,3%	96,2%	96,1%	75,0%	77,0%	68,0%
300	94,2%	97,2%	97,3%	78,0%	83,0%	75,0%
360	97,1%	98,4%	98,8%	84,0%	85,0%	80,0%
400	98,2%	98,8%	99,1%	86,0%	87,0%	82,5%
460	99,0%	99,4%	99,6%	87,0%	92,0%	85,0%
500	99,5%	99,5%	99,7%	91,0%	92,0%	87,0%
560	99,8%	99,6%	100%	92,0%	94,0%	90,0%
600	99,9%	99,6%	100%	94,0%	95,0%	92,5%
660	99,9%	99,8%	100%	95,0%	97,0%	95,0%
700	100%	100%	100%	96,0%	97,0%	96,0%
760	100%	100%	100%	96,0%	98,0%	97,0%
800	100%	100%	100%	97,0%	98,0%	97,0%
860	100%	100%	100%	97,0%	98,0%	97,0%
900	100%	100%	100%	98,0%	99,0%	98,0%
960	100%	100%	100%	99,0%	99,0%	98,0%
1000	100%	100%	100%	100%	100%	98,0%

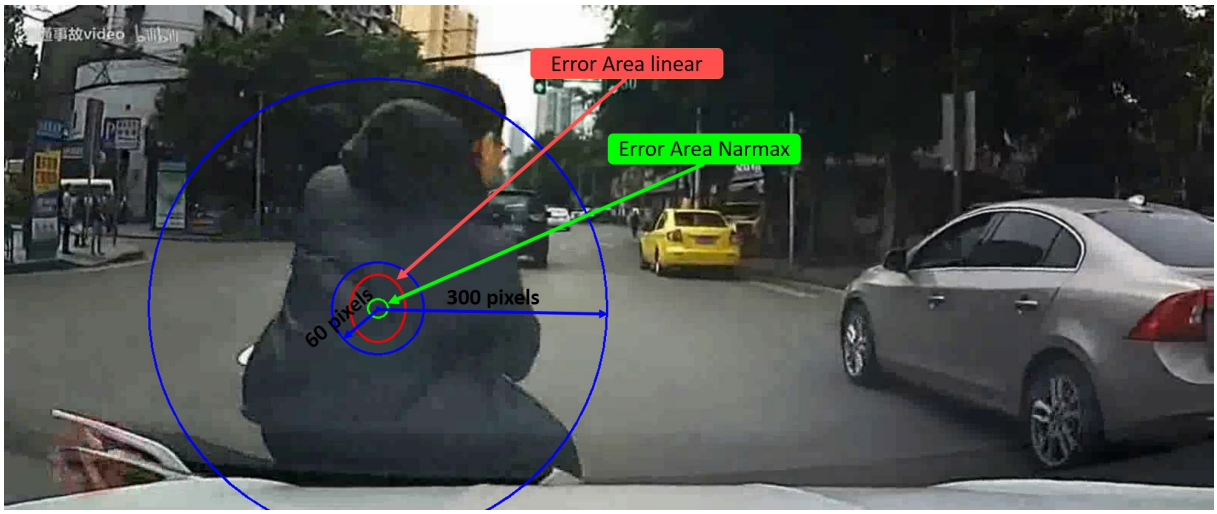


Figure 6.2.5: DADA2000 frame. Crash object circle for 60 and 300 pixels (blue). Error linear calibration area (red). Error NARMAX calibration area (green).

argue to use, even though they also argue every video was watched by at least two people, indicating that not all of were watched by the 5 users.

For a 300 pixels threshold we have obtained more than 90 % of success rate for mid and end sections of the accident event. This test shows worse results on the start sections of the accident but near the results claimed by the authors of the dataset. Mid and end sections show good results even highly than the claimed by the authors.

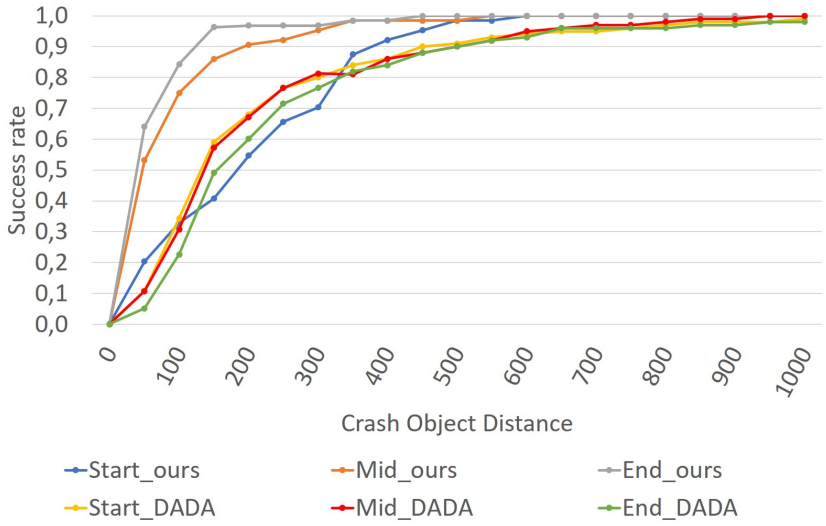
Results shows that early accident detection is worse than for the middle partition because users take some time to react. In the last part of the accident the users that have recorded this test using NARMAX calibration unlike using the method of linear calibration have kept their eyes on the accident.

Figure 6.2.6 shows the success rate and frame ratio curves, obtained for a threshold between 60 and 1000 pixels and for the three sub-sections presented in the accident partition, in the same way that the shown by the dataset authors in [2]. Our results outperform the obtained for our baseline for all the analysed sub-sections except the Success rate start section which has a lower curve in some sections of the curve (160-400 pixels).

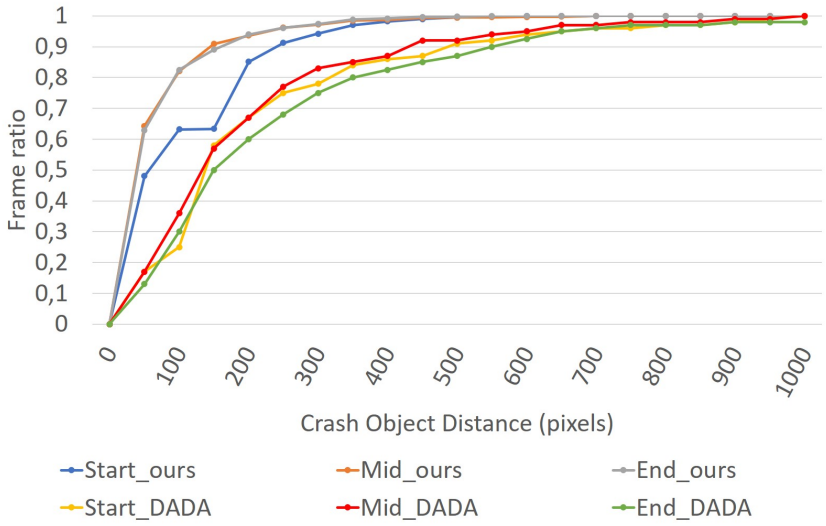
6.2.6.2 Heat attention map on DADA2000

Heat attention map is the visualization tool used for this method too as done with the linear calibration. For this purpose in necessary to establish a temporal window which has been set to 1 second as done in the DR(eye)VE project, this window represented 60 measures using the linear calibration but as this method works better with 30 frames per second this window has been set to 30 measures.

Figure 6.2.7 shows the way this experiment has been done. The user is looking at the



(a) Success rate obtained by the system proposed using NARMAX calibration.



(b) Frame ratio obtained by the system proposed using NARMAX calibration.

Figure 6.2.6: Results obtained on the testing using NARMAX calibration.

accident while the camera is recording him. The attention map generated by the samples is shown. As we can see, our attention map is included in the baseline map, showing the usability of our tool but using a cheaper and non-intrusive sensor than the authors of the dataset.

Figure 6.2.8 shows how temporal aggregation on heat attention map can help to predict where the accident will happen. Displayed accidents involved other vehicles were the ego-vehicle is an observer of the accident. This figure shows the heat map created by 4 users. The figure shows 2 accident situations with their corresponding maps in the image where the accident takes place as a heat map. Images displayed are with a temporal difference of 1 second. As we can see, the heat map is able to correctly predict all the crash objects.

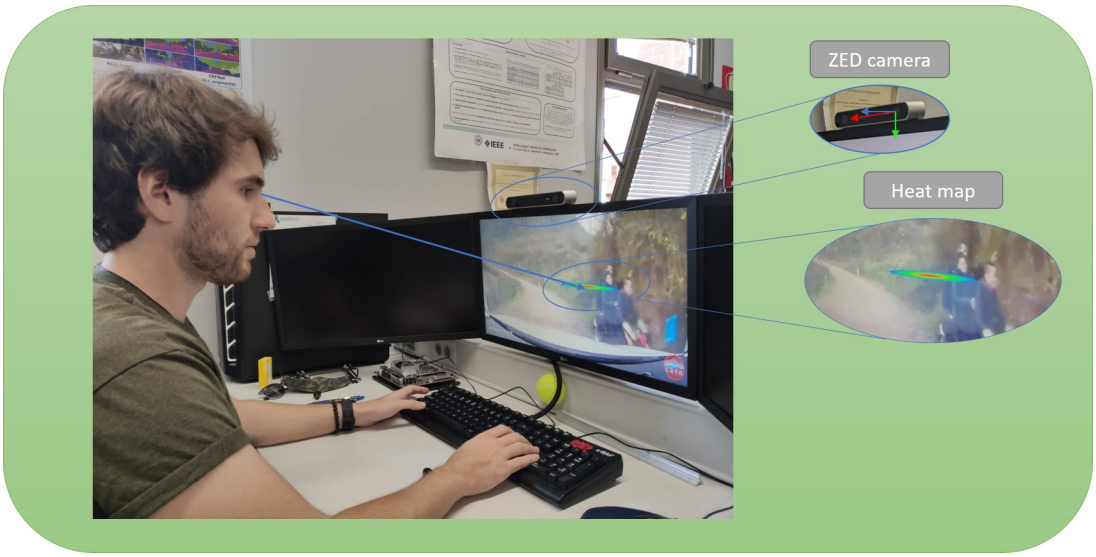


Figure 6.2.7: Heat attention map captured on real time during a DADA test using NARMAX calibration method.

Being able to generate attention map is the key of this work in order to get where human looks when they are driving because this will be the future to understand how they drive.

Figure 6.2.8: Attention map with temporal aggregation for the frame t, t-1 s, t-2 s and t-3 s using NARMAX calibration method.

Chapter 7

Conclusion and future works

7.1 Conclusion

This work shows our first version of our Vehicle to User layer in our Tech4AgeCar project. For this purpose two systems has been developed. Firstly, a HMI (Human Machine Interface) that currently is working on the Autonomous vehicle developed by the Robesafe group at the University of Alcalá, which has been developed using the ROS ecosystem to integrate it with the other layers implemented on the car, as was presented in chapter 2.

Secondly a driver gaze focalization method based on gaze tracker using the OpenFacet tool has been implemented. Different tests have been done in order to prove the accuracy of this tool, also two different calibration methods have been implemented in order to get better the accuracy of the tool. Different camera parameters have been tested, as well as the position of the camera to know which is the optimal position in the real vehicle. Also, the system has been prove to work with glasses and an extended precision test has been done with some users.

Finally the tool has been validated in the challenging public dataset DADA2000, which collects accidents videos annotated to test driver attention, compared with ours, showing on par results with our base line in similar conditions. This fact confirms our technique is a good tool for driver attention monitoring able to be used in the design of taking over systems and driving environment awareness systems for automated vehicles.

Results showing the two different methods are presented. Results are quite good and show that this method based on computer vision is a good alternative to expensive and intrusive methods using active desktop-mounted eye-trackers systems.

7.2 Future works

This works represents the beginning of our future Vehicle to User layer for our autonomous car.

Our HMI will be in continuous developing showing different tools to the user that helps him to obtain more information about the car and the surrounding environment. Depending of the moment and the driver some parameters will be shown and some other will not because the HMI will learn about the information the user needs to understand the autonomous navigation task, and the data the autonomous navigation needs from the user in case its reliability was low.

The gaze focalization unit will be used in future works to develop a dataset of attention map using the heat map generated by the tool and the actions done by the driver to drive the vehicle in a powerful simulator or even using the instrumented vehicle. With this dataset a visual attention model for autonomous driving will be generated using deep learning to learn how the user drive, as a function of what he is looking at. All this information will serve to translate the human decision making actions to our autonomous navigation architecture, reaching an interpretability of the autonomous navigation actions

Bibliography

[1] T. Baltrusaitis, A. Zadeh, Y. C. Lim, and L.-P. Morency, "Openface 2.0: Facial behavior analysis toolkit," in *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. IEEE, 2018, pp. 59–66.

[2] J. Fang, D. Yan, J. Qiao, J. Xue, H. Wang, and S. Li, "Dada-2000: Can driving accident be predicted by driver attentionÆ analyzed by a benchmark," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 4303–4309.

[3] "The history and evolution of self-driving cars," 2020, https://kambria.io/blog/the-history-and-evolution-of-self-driving-cars/.

[4] P. Tientrakool, Y.-C. Ho, and N. F. Maxemchuk, "Highway capacity benefits from using vehicle-to-vehicle communication and sensors for collision avoidance," in *2011 IEEE Vehicular Technology Conference (VTC Fall)*. IEEE, 2011, pp. 1–5.

[5] F. Jimenez, *Intelligent Vehicles: Enabling technologies and future developments*. Butterworth-Heinemann, 2017.

[6] L. Yang, K. Dong, A. J. Dmitruk, J. Brighton, and Y. Zhao, "A dual-cameras-based driver gaze mapping system with an application on non-driving activities monitoring," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[7] E. Dalmaijer, S. MathÃ´t, and S. Stigchel, "Pygaze: An open-source, cross-platform toolbox for minimal-effort programming of eyetracking experiments," *Behavior Research Methods*, vol. 46, 11 2013.

[8] M. Cognolato, M. Atzori, and H. Müller, "Head-mounted eye gaze tracking devices: An overview of modern devices and recent advances," *Journal of rehabilitation and assistive technologies engineering*, vol. 5, p. 2055668318773991, 2018.

[9] J. Shen, S. Zafeiriou, G. G. Chrysos, J. Kossaifi, G. Tzimiropoulos, and M. Pantic, "The first facial landmark tracking in-the-wild challenge: Benchmark and results," in *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, Dec 2015, pp. 1003–1011.

[10] Y. Xia, D. Zhang, J. Kim, K. Nakayama, K. Zipser, and D. Whitney, "Predicting driver attention in critical situations," in *Asian conference on computer vision*. Springer, 2018, pp. 658–674.

[11] N. Mizuno, A. Yoshizawa, A. Hayashi, and T. Ishikawa, "Detecting driver's visual attention area by using vehicle-mounted device," in *2017 IEEE 16th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*. IEEE, 2017, pp. 346–352.

[12] F. Vicente, Z. Huang, X. Xiong, F. De la Torre, W. Zhang, and D. Levi, "Driver gaze tracking and eyes off the road detection system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 2014–2027, 2015.

[13] R. A. Naqvi, M. Arsalan, G. Batchuluun, H. S. Yoon, and K. R. Park, "Deep learning-based gaze detection system for automobile drivers using a nir camera sensor," *Sensors*, vol. 18, no. 2, p. 456, 2018.

[14] P. Jiménez, L. M. Bergasa, J. Nuevo, N. Hernández, and I. G. Daza, "Gaze fixation system for the evaluation of driver distractions induced by ivis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1167–1178, 2012.

[15] A. Palazzi, D. Abati, F. Solera, R. Cucchiara *et al.*, "Predicting the driver's focus of attention: the dr (eye) ve project," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 7, pp. 1720–1733, 2018.

[16] P. Gershon, K. R. Sita, C. Zhu, J. P. Ehsani, S. G. Klauer, T. A. Dingus, and B. G. Simons-Morton, "Distracted driving, visual inattention, and crash risk among teenage drivers," *American journal of preventive medicine*, vol. 56, no. 4, pp. 494–500, 2019.

[17] T. Deng, K. Yang, Y. Li, and H. Yan, "Where does the driver look? top-down-based saliency detection in a traffic driving environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, pp. 2051–2062, 2016.

[18] A. Palazzi, F. Solera, S. Calderara, S. Alletto, and R. Cucchiara, "Learning where to attend like a human driver," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 920–925.

[19] A. Tawari and B. Kang, "A computational framework for driver's visual attention using a fully convolutional architecture," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 887–894.

[20] S. Alletto, A. Palazzi, F. Solera, S. Calderara, and R. Cucchiara, "Dr (eye) ve: a dataset for attention-based tasks with applications to autonomous and assisted driving," in *Proceedings of the ieee conference on computer vision and pattern recognition workshops*, 2016, pp. 54–60.

[21] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving video database with scalable annotation tooling," 2018.

[22] G. Borghi, M. Venturelli, R. Vezzani, and R. Cucchiara, "Poseidon: Face-from-depth for driver pose estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4661–4670.

[23] L. Fridman, P. Langhans, J. Lee, and B. Reimer, "Driver gaze region estimation without use of eye movement," *IEEE Intelligent Systems*, vol. 31, no. 3, pp. 49–56, 2016.

[24] A. Tawari and M. M. Trivedi, "Robust and continuous estimation of driver gaze zone by dynamic analysis of multiple face videos," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE, 2014, pp. 344–349.

[25] M. Tradacete, Á. Sáez, J. F. Arango, C. G. Huélamo, P. Revenga, R. Barea, E. López-Guillén, and L. M. Bergasa, "Positioning system for an electric autonomous vehicle based on the fusion of multi-gnss rtk and odometry by using an extented kalman filter," in *Workshop of Physical Agents*. Springer, 2018, pp. 16–30.

[26] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, Tech. Rep., 1992.

[27] J. L. Fernández, R. Sanz, J. Benayas, and A. R. Diéguez, "Improving collision avoidance for mobile robots in partially known environments: the beam curvature method," *Robotics and Autonomous Systems*, vol. 46, no. 4, pp. 205–219, 2004.

[28] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE, 2014, pp. 420–425.

[29] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[30] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[31] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "Erfnet: Efficient residual factorized convnet for real-time semantic segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 263–272, 2017.

[32] S. Chen and S. A. Billings, "Representations of non-linear systems: the narmax model," *International journal of control*, vol. 49, no. 3, pp. 1013–1032, 1989.

[33] S. Billings, M. Korenberg, and S. Chen, "Identification of non-linear output-affine systems using an orthogonal least-squares algorithm," *International Journal of Systems Science*, vol. 19, no. 8, pp. 1559–1568, 1988.

[34] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[35] A. Y. Ng, S. Gould, M. Quigley, A. Saxena, and E. Berger, "Stair: The stanford artificial intelligence robot project," 2008.

[36] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.

[37] B. Amos, B. Ludwiczuk, and M. Satyanarayanan, "Openface: A general-purpose face recognition library with mobile applications," CMU-CS-16-118, CMU School of Computer Science, Tech. Rep., 2016.

[38] T. Baltrusaitis, P. Robinson, and L.-P. Morency, "Constrained local neural fields for robust facial landmark detection in the wild," in *Proceedings of the IEEE international conference on computer vision workshops*, 2013, pp. 354–361.

[39] D. Cristinacce and T. F. Cootes, "Feature detection and tracking with constrained local models." in *Bmvc*, vol. 1, no. 2. Citeseer, 2006, p. 3.

[40] T. Baltrušaitis, P. Robinson, and L.-P. Morency, "3d constrained local model for rigid and non-rigid facial tracking," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 2610–2617.

[41] J. Peng, L. Bo, and J. Xu, "Conditional neural fields," in *Advances in neural information processing systems*, 2009, pp. 1419–1427.

[42] T. Qin, T.-Y. Liu, X.-D. Zhang, D.-S. Wang, and H. Li, "Global ranking using continuous conditional random fields," in *Advances in neural information processing systems*, 2009, pp. 1281–1288.

[43] K. Kim, T. Baltrusaitis, A. B. Zadeh, L.-P. Morency, and G. G. Medioni, "Holistically constrained local model: Going beyond frontal poses for facial landmark detection." in *BMVC*, 2016, pp. 1–12.

[44] J. Shen, S. Zafeiriou, G. G. Chrysos, J. Kossaifi, G. Tzimiropoulos, and M. Pantic, "The first facial landmark tracking in-the-wild challenge: Benchmark and results," in *Proceedings of the IEEE international conference on computer vision workshops*, 2015, pp. 50–58.

[45] B. F. Klare, B. Klein, E. Taborsky, A. Blanton, J. Cheney, K. Allen, P. Grother, A. Mah, and A. K. Jain, "Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1931–1939.

[46] S. Zhu, C. Li, C. Change Loy, and X. Tang, "Face alignment by coarse-to-fine shape searching," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4998–5006.

[47] G. Tzimiropoulos, "Project-out cascaded regression with an application to face alignment," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3659–3667.

[48] J. A. Hesch and S. I. Roumeliotis, "A direct least-squares (dls) method for pnp," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 383–390.

[49] M. La Cascia, S. Sclaroff, and V. Athitsos, "Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3d models," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 4, pp. 322–336, 2000.

[50] A. Asthana, S. Zafeiriou, S. Cheng, and M. Pantic, "Incremental face alignment in the wild," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1859–1866.

[51] J. M. Saragih, S. Lucey, and J. F. Cohn, "Deformable model fitting by regularized landmark mean-shift," *International journal of computer vision*, vol. 91, no. 2, pp. 200–215, 2011.

[52] G. Fanelli, J. Gall, and L. Van Gool, "Real time head pose estimation with random regression forests," in *CVPR 2011*. IEEE, 2011, pp. 617–624.

[53] T. Baltrušaitis, P. Robinson, and L.-P. Morency, "Openface: an open source facial behavior analysis toolkit," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016, pp. 1–10.

[54] E. Wood, T. Baltrusaitis, X. Zhang, Y. Sugano, P. Robinson, and A. Bulling, "Rendering of eyes for eye-shape registration and gaze estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3756–3764.

[55] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "Mpiigaze: Real-world dataset and deep appearance-based gaze estimation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 1, pp. 162–175, 2017.

[56] ——, "Appearance-based gaze estimation in the wild," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4511–4520.

[57] E. Wood and A. Bulling, "Eyetab: Model-based gaze estimation on unmodified tablet computers," in *Proceedings of the Symposium on Eye Tracking Research and Applications*, 2014, pp. 207–210.

[58] E. Wood, T. Baltrušaitis, L.-P. Morency, P. Robinson, and A. Bulling, "Learning an appearance-based gaze estimator from one million synthesised images," in *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, 2016, pp. 131–138.

[59] T. Baltrušaitis, M. Mahmoud, and P. Robinson, "Cross-dataset learning and person-specific normalisation for automatic action unit detection," in *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, vol. 6. IEEE, 2015, pp. 1–6.

[60] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009.

[61] S. M. Mavadati, M. H. Mahoor, K. Bartlett, P. Trinh, and J. F. Cohn, "Disfa: A spontaneous facial action intensity database," *IEEE Transactions on Affective Computing*, vol. 4, no. 2, pp. 151–160, 2013.

[62] G. McKeown, M. F. Valstar, R. Cowie, and M. Pantic, "The semaine corpus of emotionally coloured character interactions," in *2010 IEEE International Conference on Multimedia and Expo*. IEEE, 2010, pp. 1079–1084.

[63] X. Zhang, L. Yin, J. F. Cohn, S. Canavan, M. Reale, A. Horowitz, P. Liu, and J. M. Girard, "Bp4d-spontaneous: a high-resolution spontaneous 3d dynamic facial expression database," *Image and Vision Computing*, vol. 32, no. 10, pp. 692–706, 2014.

[64] P. Lucey, J. F. Cohn, K. M. Prkachin, P. E. Solomon, and I. Matthews, "Painful data: The unbc-mcmaster shoulder pain expression archive database," in *Face and Gesture 2011*. IEEE, 2011, pp. 57–64.

[65] A. Savran, N. Alyüz, H. Dibeklioğlu, O. Çeliktutan, B. Gökberk, B. Sankur, and L. Akarun, "Bosphorus database for 3d face analysis," in *European Workshop on Biometrics and Identity Management*. Springer, 2008, pp. 47–56.

[66] M. F. Valstar, B. Jiang, M. Mehu, M. Pantic, and K. Scherer, "The first facial expression recognition and analysis challenge," in *Face and Gesture 2011*. IEEE, 2011, pp. 921–926.

[67] T. Baltrušaitis, P. Robinson, and L.-P. Morency, "Continuous conditional neural fields for structured regression," in *European conference on computer vision*. Springer, 2014, pp. 593–608.

[68] J. Nicolle, K. Bailly, and M. Chetouani, "Real-time facial action unit intensity prediction with regularized metric learning," *Image and Vision Computing*, vol. 52, pp. 1–14, 2016.

[69] S. Kaltwang, S. Todorovic, and M. Pantic, "Latent trees for estimating intensity of facial action units," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 296–304.

[70] A. Gudi, H. E. Tasli, T. M. Den Uyl, and A. Maroulis, "Deep learning based facs action unit occurrence and intensity estimation," in *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, vol. 6. IEEE, 2015, pp. 1–5.

[71] K. Zhao, W.-S. Chu, and H. Zhang, "Deep region and multi-label learning for facial action unit detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3391–3399.

[72] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[73] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[74] W. Fang and T. Chang, "Calibration in touch-screen systems," *Texas Instruments Incorporated*, vol. 10, 2007.

[75] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6569–6578.

[76] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 3464–3468.

[77] G. A. Mills-Tettey, A. Stentz, and M. B. Dias, "The dynamic hungarian algorithm for the assignment problem with changing costs," 2007.

[78] "Tobii pro glasses 2," 2020, https://www.tobiipro.com/product-listing/tobii-pro-glasses-2/.

[79] Z. Bylinskii, T. Judd, A. Borji, L. Itti, F. Durand, A. Oliva, and A. Torralba, "Mit saliency benchmark."

[80] A. Borji and L. Itti, "Cat2000: A large scale fixation dataset for boosting saliency research," *arXiv preprint arXiv:1505.03581*, 2015.

Appendix A

Manual de usuario

A.1 Manual git with docker

Git config

```
$ git config --global user.email "you@example.com"
$ git config --global user.name "username"
```

Problem: Permission denied (publickey). fatal: Could not read from remote repository. https://stackoverflow.com/questions/12940626/github-error-message-permission-denied-publickey

```
$ git config --get remote.origin.url
$ ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
$ eval "$(ssh-agent -s)"
$ ssh-add ~/.ssh/id_rsa
$ sudo apt-get install xclip
$ xclip -sel clip < ~/.ssh/id_rsa.pub
```

Go to https://github.com/settings/keys Go to Settings->SSH keys(Personal settings side bar)->Add SSH key->fill out form(key is on your clipboard, just use ctrl+v)->Add key

```
$ git init
$ git add .
$ git commit -m "hmi_sept"
$ git branch -M master
$ git remote add origin git@github.com:kuasho/hmi_tech4agecar.git
$ git push -u origin master
```

A.2 Manual docker preparation

Login as root.

```
$ xhost +
$ docker run -it --net host --privileged -u root -v /tmp/.X11-unix:/tmp/.X11-unix
-v $directorio -v $disco_externo -v $disco_externo2
-e DISPLAY=unix$DISPLAY $1 /bin/bash
```

Inside the docker

```
$ apt update
$ apt install sudo
$ apt install gedit
$ adduser robesafe_docker
$ usermod -aG sudo robesafe_docker
```

Commit the docker and login as robesafe_docker

```
$ xhost +
$ docker run -it --net host --privileged -u robesafe_docker
-v /tmp/.X11-unix:/tmp/.X11-unix -v $directorio -v $disco_externo
-v $disco_externo2 -e DISPLAY=unix$DISPLAY $1 /bin/bash
```

To change color of the user to difference the docker terminal

```
$ gedit ~/.bashrc
force_color_prompt=yes
PS1='${debian_chroot:+($debian_chroot)}\[\033[01;34m\]\u@\h\[\033[00m\]
:\[\    033[01;34m\]\w\[\033[00m\]\$ '
$ source ~/.bashrc
```

NVIDIA driver docker

```
$ sudo apt install kmod
$ sudo apt install module-init-tools
$ sudo ./NVIDIA-Linux-x86_64-430.26.run ?no-kernel-module
```

CUDA installation

```
$ echo "export PATH=/usr/local/cuda-10.1/bin:/usr/local/cuda-10.1/
nsight-compute-2019.4.0${PATH:+:${PATH}}" >> ~/.bashrc
$ echo "export LD_LIBRARY_PATH=/usr/local/cuda-10.1/lib64\${LD_LIBRARY_PATH:
+:${LD_LIBRARY_PATH}}"  >> ~/.bashrc
```

A.3 Manual ROS installation

```
$ sudo apt install lsb-core
```

http://wiki.ros.org/kinetic/Installation/Ubuntu

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main"
> /etc/apt/sources.list.d/ros-latest.list'
$ sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
$ sudo apt-get update
$ sudo apt-get install ros-kinetic-desktop-full
$ sudo rosdep init
$ rosdep update
$ echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
```

```
$ echo "source /home/robesafe/Javier/smartelderlycar_ws/devel/setup.bash" >> ~/.bashrc
$ echo "export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:/home/robesafe/Javier/smartelderlycar_ws"
 >> ~/.bashrc
$ source ~/.bashrc

$ sudo apt install python-rosinstall python-rosinstall-generator python-wstool
build-essential
$ sudo apt-get install ros-kinetic-geographic-msgs
$ sudo apt-get install ros-kinetic-geodesy
```

A.4 Manual Qtcreator installation

Download from https://download.qt.io/archive/qtcreator/4.0/4.0.2/qt-creator-opensource-linux-x86_64-4.0.2.run

```
$ chmod +x qt-creator-opensource-linux-x86_64-4.0.2.run
$ ./qt-creator-opensource-linux-x86_64-4.0.2.run
$ sudo ln -s /home/robesafe_docker/qtcreator-4.0.2/bin/qtcreator /usr/local/bin/qtcreator
$ sudo -s gedit /etc/profile
$ export PATH=/opt/Qt5.7.0/Tools/QtCreator/bin:$PATH
. /etc/profile
```

A.5 Openface Dependency installation

OpenFace requires cmake, OpenCV 4.0.0 (or newer), OpenBLAS, dlib, C++17 compiler (tbb and boost are optional additional dependencies that will be used if present).

If you already have any of the following dependencies you can skip those steps

A.5.1 Get newest GCC, done using:

```
$ sudo apt-get update
$ sudo apt-get install build-essential
$ sudo add-apt-repository ppa:ubuntu-toolchain-r/test -y
$ sudo apt-get -y update
$ sudo apt-get install g++-8
```

A.5.2 Cmake:

```
$ sudo apt-get --purge remove cmake-qt-gui -y
$ sudo apt-get --purge remove cmake -y
$ sudo mkdir -p cmake_tmp
$ cd cmake_tmp
$ sudo wget https://cmake.org/files/v3.10/cmake-3.10.1.tar.gz
$ sudo tar -xzvf cmake-3.10.1.tar.gz -qq
$ cd cmake-3.10.1/
```

```
$ sudo ./bootstrap
$ sudo make -j8
$ sudo make install
$ cd ../..
$ sudo rm -rf cmake_tmp
```

A.5.3 Get OpenBLAS:

```
$ sudo apt-get install libopenblas-dev
```

A.5.4 Download and compile OpenCV 4.1.0

A.5.4.1 Install OpenCV dependencies:

```
$ sudo apt-get install git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev li
$ sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev
libpng-dev libtiff-dev libdc1394-22-devbswscale-dev
```

A.5.4.2 Download OpenCV 4.1.0

From https://github.com/opencv/opencv/archive/4.1.0.zip

```
$ sudo wget https://github.com/opencv/opencv/archive/4.1.0.zip
```

A.5.4.3 Unzip it and create a build folder:

```
$ sudo unzip 4.1.0.zip
$ cd opencv-4.1.0
$ sudo mkdir build
$ cd build
```

A.5.4.4 Build it using:

```
$ sudo cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local
-D BUILD_TIFF=ON -D WITH_TBB=ON ..
$ sudo make -j8
$ sudo make install
```

A.5.4.5 Download and compile dlib:

```
$ sudo wget http://dlib.net/files/dlib-19.13.tar.bz2;
$ sudo tar xf dlib-19.13.tar.bz2;
$ cd dlib-19.13;
$ sudo mkdir build;
$ cd build;
$ sudo cmake ..;
```

```
$ sudo cmake --build . --config Release;
$ sudo make install;
$ sudo ldconfig;
$ cd ../..;
```

A.5.4.6 Get Boost (optional):

```
$ sudo apt-get install libboost-all-dev
```

A.6 Actual OpenFace installation

A.6.1 Get OpenFace

```
$ git clone https://github.com/TadasBaltrusaitis/OpenFace.git
```

A.6.2 Create an out-of-source build directory to store the compiled artifacts:

```
$ cd OpenFace
$ mkdir build
$ cd build
```

A.6.3 Compile the code using (if using g++ version 8, change to clang or other version appropriately):

```
$  cmake -D CMAKE_CXX_COMPILER=g++-8 -D CMAKE_C_COMPILER=gcc-8 -D CMAKE_BUILD_TYPE=RELEASE ..
 make
```

A.6.4 Test it with

A.6.4.1 For videos:

```
$  ./bin/FaceLandmarkVid -f "../samples/changeLighting.wmv" -f "../samples/2015-10-15-15-14.avi"
```

A.6.4.2 For images:

```
$  ./bin/FaceLandmarkImg -fdir "../samples/" -wild
```

A.6.4.3 For multiple faces in videos:

```
$  ./bin/FaceLandmarkVidMulti -f ../samples/multi_face.avi
```

A.6.4.4 For feature extraction (facial landmarks, head pose, AUs, gaze and HOG and similarity aligned faces):

```
$  ./bin/FeatureExtraction -verbose -f "../samples/default.wmv"
```

A.7 Carla installation

A.7.1 Unreal Installation

A.7.1.1 Dependencies

```
$  sudo apt-get update &&
$  sudo apt-get install wget software-properties-common &&
$  sudo add-apt-repository ppa:ubuntu-toolchain-r/test &&
$  wget -O - https://apt.llvm.org/llvm-snapshot.gpg.key|sudo apt-key add - && sudo
apt-add-repository "deb http://apt.llvm.org/xenial/ llvm toolchain-xenial-8 main"
&& sudo apt-get update
```

A.7.1.2 Dependencies for Ubuntu 18.04

```
$  sudo apt-get install build-essential clang-8 lld-8 g++-7 cmake ninja-build libvulkan1 python
python-pip python-dev python3-dev python3-pip libpng-dev libtiff5-dev libjpeg-dev tzdata sed
curl  unzip autoconf libtool rsync libxml2-dev libxerces-c-dev && pip2 install
--user setuptools && pip3 install --user -Iv setuptools==47.3.1 && pip2 install
--user distro && pip3 install --user distro
```

A.7.1.3 UnrealEngine 4.24

```
$  git clone --depth=1 -b 4.24 https://github.com/EpicGames/UnrealEngine.git ~/UnrealEngine_4.24
$  cd ~/UnrealEngine_4.24
$  wget
https://carla-releases.s3.eu-west-3.amazonaws.com/Linux/UE_Patch/430667-13636743-patch.txt
 ~/430667-13636743-patch.txt patch --strip=4 < ~/430667-13636743-patch.txt
$  ./Setup.sh && ./GenerateProjectFiles.sh && make
```

A.7.2 Carla Installation

```
$  sudo apt-get install aria2
$  git clone https://github.com/carla-simulator/carla
$  cd carla/
$  git checkout c18ccca7
```

```
$  cd carla/
$  ./Update.sh
$  export UE4_ROOT=~/UnrealEngine_4.24
$  make launch
$  make PythonAPI
$  make package
$  make clean
```

A.7.3 ROS bridge Installation

```
$  mkdir -p ~/carla-ros-bridge/catkin_ws/src
$  cd ~/carla-ros-bridge
$  git clone https://github.com/carla-simulator/ros-bridge.git
$  cd ros-bridge
$  git submodule update --init
$  cd ../catkin_ws/src
$  ln -s ../../ros-bridge
$  source /opt/ros/melodic/setup.bash
$  cd ..

$  rosdep update
$  rosdep install --from-paths src --ignore-src -r

$  catkin_make
```

A.8 Run Carla with ROS

```
$  ./carla/Dist/CARLA_Shipping_0.9.9.4-113-gc18ccca7/LinuxNoEditor/CarlaUE4.sh

$  source ~/carla-ros-bridge/catkin_ws/devel/setup.bash
$  roslaunch carla_ros_bridge carla_ros_bridge_with_example_ego_vehicle.launch
```

Appendix B

Specifications

This appendix details the main hardware and software used in this master thesis.

B.1 Hardware

- Laboratory desktop PC
  - Intel i7 9700k
  - Nvidia 2080 ti 11 GB
  - 32 GB DDR4 Ram
  - 500 GB SSD
- Home desktop PC
  - Ryzen 7 3700X
  - Nvidia 1660 ti 6 GB
  - 16 GB DDR4 Ram
  - 500 GB SSD
- Vehicle laptop (MSI GT62VR-7RE i7-7700HQ)
  - Intel i7 7700HQ
  - Nvidia 1070 8 GB
  - 16 GB DDR4 Ram
  - 500 GB SSD
- Stereo Labs ZED camera

- Vehicle Raspberry Pi 2B+
- Nvidia Jetson Nano
- 14 inch screen
- 15 inch touch screen

B.2 Software

- Operative system GNU/Linux Ubuntu 18.04.3 LTS (Bionic Beaver)
- Code editor Visual Studio Code
- Docker 19.03
- ROS Melodic
- ROS Kinetic
- Carla Simulator 0.9.9.5
- Texmaker editor LaTeX

Appendix C

Budget

This appendix details the main hardware and software used in this master thesis.

C.1 Hardware

This section will detail the budget for the hardware used on this master thesis.

Table C.1: Desktop home PC.

Desktop home PC		
Components	Name	Cost
Processor	ryzen 7 3700x	364,00 €
GPU	Nvidia 1660 ti	300,00 €
RAM	16 GB 3200 Mhz	90,00 €
SSD	Samsung 500 GB	77,00 €
Other hardware		250,00 €
Total		1.081,00 €

Table C.2: Desktop laboratory PC

Desktop laboratory PC		
Components	Name	Cost
Processor	Intel i7 9700k	350,00 €
GPU	Nvidia 2080 ti	1.300,00 €
RAM	32 GB 3200 Mhz	180,00 €
SSD	Samsung 500 GB	77,00 €
Other hardware		400,00 €
Total		2.307,00 €

Table C.3: Embedded systems

Embedded systems	
Name	Cost
Raspberry Pi 2 B+	40,00 €
Nvidia Jetson Nano	100,00 €

Table C.4: Car hardware

Car hardware	
Name	Cost
Screen	115,00 €
Touch screen	200,00 €

C.2 Software

This section will detail the budget for the software used on this master thesis.

Table C.5: Software

Software	Cost
Ubuntu 18.04	- €
Ubuntu 16.04	- €
ROS Melodic	- €
ROS Kinetic	- €
Matlab	- €
Carla simulator	- €
QtCreator	- €
Visual Studio Code	- €
Latex editor	- €
Microssoft Office	- €

C.3 Professional fees

This section will detail the budget for the Professional fees used on this master thesis.

Table C.6: Professional fees

Professional fees	Months	Cost/month (TAX included)	Total cost
Engineering	9	1400 €	12.600 €
Typing	1	1400 €	1.400 €
Total			14.000 €

C.4 Total cost

Table C.7: Total cost

Item	Amortization	Total cost
Desktop home PC	4 years	270,25 €
Desktop laboratory PC	4 years	576,75 €
Embedded systems		140,00 €
Car hardware		140,00 €
Software		- €
Professional fees		14.000,00 €
Total cost		15.127,00 €

C.5 General expenses and industrial profit

Following the PEM 2020 classification, overheads are estimated at 13% of total running costs, while industrial profit is around 6% of the total execution costs.

Table C.8: General expenses and industrial profit

Execution costs	Cost
Material execution costs	15.127,00 €
General expenses (13%)	1.966,51 €
Industrial profit (6%)	907,62 €
Execution budget per contract	18.001,13 €

C.6 Total amount of the budget

The final budget for this Master's Thesis ends with the application of VAT, stipulated in 21%, based on the execution budget per contract.

Table C.9: Total amount of the budget

Item	Cost
Execution budget	18.001,13 €
VAT (21%)	3.780,24 €
Total Cost	21.781,37 €

The final estimated budget for the completion of this Master's Thesis amounts to a total of 21.781,37 €(twenty one thousand seven hundred and eighty one euros and thirty seven cents).

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá